SURFING ON THE BACKDOOR WAVE

Advances and Future Directions in Backdoor Attacks and Defenses in AI



STEFANOS KOFFAS

Propositions

accompanying the dissertation

Surfing on the Backdoor Wave

Advances and Future Directions in Backdoor Attacks and Defenses in AI

by

Stefanos Koffas

1. Comprehensive trigger stealthiness leads to harder to defend backdoor attacks (from this thesis).
2. Backdoor attacks are not constrained by the application modality (from this thesis).
3. A common evaluation framework is missing from the backdoor defense literature (from this thesis).
4. In the next five years, agentic systems will amplify cyber risk but they will result in more resilient systems in the long run.
5. Engineering-heavy papers are valuable research contributions when they enable reproducible evaluation or reveal previously unknown aspects of existing methods.
6. LLMs do not make people stupid, laziness does.
7. Open-weight AI models are not equivalent to open-source software.
8. Whether AI models understand the task they are doing is irrelevant.
9. Europe should abolish seasonal clock changes and keep a single, permanent time standard.
10. A foldable bike is more useful than a normal-sized bike in the Netherlands.

These propositions are regarded as opposable and defendable, and have been approved as such by the promotor(s) Prof.dr. G. Smaragdakis, Prof.dr. M. Conti, and Prof.dr. S. Picek.

Surfing on the Backdoor Wave

Advances and Future Directions in Backdoor Attacks and Defenses in AI

SURFING ON THE BACKDOOR WAVE

ADVANCES AND FUTURE DIRECTIONS IN BACKDOOR ATTACKS AND DEFENSES IN AI

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus Prof.dr.ir. H. Bijl
chair of the Board for Doctorates
to be defended publicly on
Tuesday 22, September 2026 at 10:00

by

Stefanos KOFFAS

This dissertation has been approved by the promotors.

Composition of the doctoral comittee:

Rector Magnificus,	chairperson
Prof. dr. G. Smaragdakis,	Delft University of Technology, NL, *promotor*
Prof. dr. M. Conti,	University of Padua, Italy, Örebro University, Sweden, *promotor*
Prof. dr. S. Picek,	University of Zagreb, Croatia, Radboud University, NL, *promotor*
Independent Members:	
Prof. dr. M.J.G. van Eeten	Delft University of Technology, NL
Prof. dr. L. Batina	Radboud University, NL
Prof. dr. Z. Li	Shandong University, China
Dr. ir. M. Pintor	University of Cagliari, Italy
Prof. dr. O. E. Scharenborg	Delft University of Technology, NL, reserve member



Keywords: Backdoor Attacks, Backdoor Defenses, Data Poisoning

Printed by: ProefschriftMaken

Cover by: ChatGPT & Gemini & Georgina Savvaki & Stefanos Koffas.

Copyright © 2026 by S. Koffas

ISBN 978-94-6518-426-5

An electronic version of this dissertation is available at: http://repository.tudelft.nl.

CONTENTS

Summary xiii

Samenvatting xv

I Introduction & Preliminaries 1

1 Introduction 3
1.1 Historical Context . . . 3
1.2 Adversarial ML . . . 5
1.3 Motivation . . . 7
1.4 Research Questions . . . 8
1.5 Thesis Outline . . . 9
1.5.1 Backdoors in Speech . . . 9
1.5.2 Backdoors in Different Modalities . . . 10
1.5.3 Backdoor Attacks and Defenses in Depth . . . 11
1.5.4 Summary . . . 13
1.5.5 Final Insights . . . 13
1.6 Contributions . . . 13
1.6.1 About this Thesis . . . 14

2 Background 21
2.1 Machine Learning . . . 21
2.2 Deep Learning . . . 22
2.2.1 Convolutional Neural Networks . . . 23
2.3 Federated Learning . . . 25
2.4 Graph Neural Networks . . . 26
2.4.1 Applications . . . 27
2.5 Backdoor Attacks in Deep Learning . . . 28

II Backdoors in Speech 33

3 Backdoors via Ultrasonic Trigger 35
3.1 Introduction . . . 36
3.2 Preliminaries . . . 37
3.2.1 Automatic Speech Recognition . . . 37
3.2.2 Backdoor Attacks . . . 37
3.3 Inaudible Backdoor Attack . . . 37
3.3.1 Trigger . . . 37
3.3.2 Threat Model . . . 38

3.4 Experimental Setup . . . 38
3.4.1 Dataset and Input Features . . . 38
3.4.2 Neural Network Architectures . . . 38
3.5 Experimental Results . . . 39
3.5.1 Effect on Clean Accuracy . . . 40
3.5.2 On the Effect of Trigger Properties on ASR . . . 41
3.5.3 Attack on Android Application . . . 43
3.5.4 General Observations . . . 45
3.6 Conclusions and Future Work . . . 46

4 Backdoors through Stylistic Transformations 49
4.1 Introduction . . . 50
4.2 Background . . . 50
4.3 Style Backdoor Attack . . . 51
4.3.1 Threat Model . . . 51
4.3.2 Attack Formulation . . . 51
4.4 JingleBack Design . . . 52
4.4.1 Stylistic Generation . . . 52
4.4.2 Experimental Settings . . . 53
4.5 Experimental Results . . . 54
4.5.1 Effect on Clean Accuracy . . . 54
4.5.2 Backdoor . . . 54
4.5.3 Ablation Study . . . 56
4.5.4 Evasion . . . 56
4.6 Generalization on Different Datasets . . . 56
4.7 Conclusions and Future Work . . . 57

5 Emoback: Backdoors Using Emotional Prosody 59
5.1 Introduction . . . 60
5.2 Background . . . 61
5.2.1 Speaker Recognition . . . 61
5.2.2 Stage-Wise vs. End-to-End Architectures . . . 61
5.2.3 Backdoor Attacks . . . 61
5.3 Threat Model . . . 62
5.4 EmoBack . . . 62
5.5 Experimental Setup . . . 63
5.5.1 Datasets . . . 63
5.5.2 Neural Network Training . . . 64
5.5.3 Evaluation Metrics . . . 64
5.5.4 Defense Setup . . . 64
5.6 Results and Discussion . . . 66
5.6.1 Attack Performance . . . 66
5.6.2 Pruning . . . 69
5.6.3 STRIP-ViTA . . . 72
5.6.4 Pre-processing-based Defense Strategies . . . 75
5.7 Conclusions and Future Work . . . 78

III Backdoors In Different Modalities **83**

6 Backdoors in Federated Graph Neural Networks **85**
6.1 Introduction . . . 86
6.2 Background . . . 87
6.2.1 Federated Learning . . . 87
6.2.2 Graph Neural Networks . . . 88
6.2.3 Backdoor Attacks on Federated Learning . . . 88
6.3 Problem Formulation . . . 89
6.3.1 Overview . . . 89
6.3.2 Threat Model . . . 89
6.4 Backdoor Attacks against Federated GNNs . . . 91
6.4.1 General Framework . . . 91
6.4.2 Backdoored Data Generation . . . 94
6.5 Experiments . . . 95
6.5.1 Experimental Setting . . . 95
6.5.2 Backdoor Attack Results . . . 96
6.5.3 Clean Accuracy Drop . . . 103
6.6 Analysis of Backdoor Hyperparameters . . . 104
6.7 Defense . . . 106
6.8 Related Work . . . 109
6.9 Conclusions and Future Work . . . 109

7 CatBack: Universal Backdoors on Tabular Data **115**
7.1 Introduction . . . 116
7.2 Background . . . 117
7.2.1 Backdoor Attacks . . . 117
7.2.2 Characteristics of Tabular Data . . . 118
7.2.3 Machine Learning Models for Tabular Data . . . 118
7.3 Threat Model . . . 119
7.3.1 Attacker's Capabilities and Objectives . . . 119
7.4 Converting Categorical Features . . . 120
7.4.1 Primary Frequency-based Mapping . . . 120
7.4.2 Adaptive Δr Selection . . . 120
7.4.3 Identifying and Resolving Ties . . . 121
7.4.4 Final Numerical Representation . . . 121
7.4.5 Reverse Mapping . . . 121
7.4.6 Example of Our Encoding and Attack Algorithm . . . 122
7.5 Attack Methodology . . . 123
7.5.1 Initial Model Training . . . 123
7.5.2 Selection of Non-target Samples . . . 123
7.5.3 Confidence-based Sample Ranking . . . 124
7.5.4 Definition of the Backdoor Trigger . . . 124
7.5.5 Optimization of the Trigger Pattern . . . 124
7.5.6 Construction of the Poisoned Dataset . . . 125
7.5.7 Training the Backdoored Model . . . 126

7.5.8 Deployment and Attack Activation . . . 126
7.5.9 Hyperparameter Considerations . . . 126
7.5.10 Black Box Attack . . . 127
7.6 Evaluation and Results . . . 127
7.6.1 Environment and System Specification . . . 127
7.6.2 Evaluation Metrics . . . 128
7.6.3 Benign Accuracies . . . 128
7.6.4 Attack Results . . . 128
7.6.5 Detailed Attack Results . . . 130
7.6.6 Baselines . . . 130
7.6.7 Additional Threat Scenario . . . 133
7.6.8 Evaluation on Vertex AI . . . 134
7.7 Evaluation Against Defenses . . . 134
7.7.1 Spectral Signatures . . . 135
7.7.2 Neural Cleanse . . . 136
7.7.3 Beatrix . . . 137
7.7.4 Fine Pruning . . . 138
7.7.5 Evaluation with the Outlier Detection Algorithms . . . 138
7.8 Ablation Studies . . . 139
7.8.1 Partial Access to Training Data . . . 139
7.8.2 Frequency Mappings . . . 140
7.8.3 Performance Under Distribution Shifts . . . 141
7.9 Limitations and Discussion . . . 143
7.10 Related Work . . . 144
7.11 Ethical Considerations . . . 145
7.12 Conclusions and Future Work . . . 145

8 A Systematic Evaluation of Backdoor Attacks in Various Domains 151
8.1 Introduction . . . 152
8.2 Background . . . 153
8.2.1 Computer Vision . . . 153
8.2.2 Natural Language Processing . . . 153
8.2.3 Graph Data . . . 153
8.2.4 Backdoor Attacks . . . 154
8.3 Methodology . . . 157
8.3.1 Threat Model . . . 157
8.3.2 Image Classification . . . 158
8.3.3 Natural Language Processing . . . 158
8.3.4 Speech Recognition . . . 159
8.3.5 Graph Data . . . 159
8.4 Experimental Results . . . 161
8.4.1 Image Classification . . . 161
8.4.2 Natural Language Processing . . . 164
8.4.3 Speech Recognition . . . 166
8.4.4 Graph Data . . . 169
8.4.5 General Observations . . . 172

8.5 Conclusions . . . 174

9 Evaluating Backdoor Attacks and Defenses in Different Domains 181
9.1 Motivation and Significance . . . 182
9.2 Backdoor Attacks on Neural Networks . . . 184
9.2.1 Metrics . . . 184
9.3 Software Description . . . 184
9.4 Illustrative Example . . . 186
9.4.1 Dataset Selection . . . 186
9.4.2 Choosing the Attack and Defense . . . 187
9.4.3 Metrics Configuration . . . 187
9.4.4 Results . . . 188
9.5 Impact . . . 190
9.6 Limitations and Future Work . . . 190
9.7 Conclusion . . . 190

IV Backdoor Attacks and Defenses in Depth 197

10 BAN: Detecting Backdoors Activated by Adversarial Neuron Noise 199
10.1 Introduction . . . 200
10.2 Backdoors . . . 202
10.3 BAN Method . . . 203
10.3.1 The Pipeline of Training Backdoor Models . . . 203
10.3.2 Threat Model . . . 203
10.3.3 Detection with Neuron Noise . . . 204
10.3.4 Improving BTI-DBF . . . 205
10.3.5 Backdoor Defense . . . 206
10.4 Experimental Setup . . . 206
10.4.1 Datasets . . . 206
10.4.2 Models . . . 207
10.4.3 Attack Baselines . . . 207
10.4.4 Defense Baselines . . . 207
10.4.5 BAN Settings . . . 208
10.5 Experimental Results . . . 209
10.5.1 The Performance of Backdoor Detection . . . 209
10.5.2 The Performance of Backdoor Defense . . . 211
10.5.3 Defense against All-To-All Attacks . . . 213
10.5.4 Evaluation under Adaptive Attack . . . 213
10.5.5 Analysis on Prominent Features . . . 214
10.5.6 Different Poisoning Rates . . . 214
10.5.7 Against Backdoors on Additional Architectures . . . 214
10.5.8 Choosing the λ_1 . . . 214
10.5.9 Clean Label Attack . . . 216
10.6 Discussion . . . 216
10.7 Conclusions and Future Work . . . 217

11 Towards Backdoor Stealthiness in Model Parameter Space 221
11.1 Introduction . . . 223
11.2 Background & Related Work. . . . 225
11.2.1 Preliminaries on Backdoor Training . . . 225
11.2.2 Backdoor Attacks . . . 226
11.2.3 Backdoor Defenses. . . . 227
11.3 Comprehensive Backdoor Stealthiness . . . 229
11.3.1 Threat Model . . . 229
11.3.2 Lack of Parameter-Space Stealthiness . . . 229
11.3.3 `Grond` for Comprehensive Stealthiness. . . . 230
11.4 Experimental Evaluation . . . 233
11.4.1 Experimental Setup . . . 233
11.4.2 Main Results on Backdoor Mitigation . . . 235
11.4.3 Adaptive Defenses . . . 237
11.4.4 Backdoor Analysis . . . 238
11.4.5 ABI Improves Common Backdoor Attacks . . . 239
11.4.6 Backdoor Detection . . . 240
11.4.7 Ablation Study . . . 241
11.4.8 Different Architectures with Different Surrogate Models . . . 244
11.4.9 Visualization . . . 244
11.5 Stronger Defenders and Additional Analysis . . . 245
11.5.1 Proactive Defense . . . 245
11.5.2 Detection of backdoor input. . . . 245
11.5.3 Large and Transformer-Based Architectures . . . 246
11.5.4 Backdoor analysis by feature space inversion . . . 247
11.5.5 Adversarial Backdoor Injection Does Not Impact Backdoor Effectiveness in Case of No Defense. . . . 248
11.5.6 Further TAC analysis . . . 248
11.5.7 Adversarial Training as A Defense . . . 249
11.5.8 Weight Changes After FT-SAM with 10 Baseline Attacks . . . 250
11.5.9 Attempts to Craft Inputs that Minimize Changes in the Parameter Space . . . 250
11.6 Examples of Poisoned Images . . . 250
11.7 Conclusions & Future Work . . . 251

12 SoK: The Last Line of Defense: On Backdoor Defense Evaluation 257
12.1 Introduction . . . 258
12.2 Background . . . 259
12.2.1 Backdoor Attacks . . . 259
12.2.2 Backdoor Defenses. . . . 261
12.2.3 Common Threat Models . . . 262
12.3 Literature Study . . . 263
12.3.1 Methodology. . . . 263
12.3.2 Analysis of the Collected Data . . . 265
12.4 Experimental Setup . . . 272

12.5 Experimental Results . . . 274
12.5.1 Discussion on Sustainability . . . 284
12.6 Guidelines for Experimental Evaluation . . . 284
12.7 Limitations . . . 285
12.8 Related Work . . . 285
12.9 Conclusions & Future Work . . . 287

V Final Insights **313**

13 Discussion **315**
13.1 Audio Backdoors . . . 315
13.2 Non-vision Backdoors . . . 316
13.3 Comprehensive Stealthiness . . . 318
13.4 Standardized Defense Evaluation . . . 319
13.5 Limitations . . . 320
13.6 Discussion & Future work . . . 321
13.6.1 Real-world Backdoor Feasibility . . . 321
13.6.2 Security Critical Domains . . . 321
13.6.3 Bridging the Gap Between Research and Industry . . . 322
13.6.4 Standardized and Reproducible Evaluation . . . 322
13.6.5 Backdoor Threats in Foundation Models . . . 323
13.6.6 Backdoors, Observability, and Interpretability . . . 323
13.6.7 Connecting Backdoors to LLM Jailbreaking . . . 323
13.7 Conclusions . . . 324

Curriculum Vitæ **325**

List of Publications **327**

Acknowledgements **331**

SUMMARY

Deep neural networks (DNNs) have become central to a wide range of modern applications, including systems used in security- and safety-critical settings. As these models are increasingly deployed in practice, they have also become attractive targets for adversaries. Among the most concerning threats are backdoor attacks, in which an attacker implants a hidden malicious functionality into a model. This functionality is activated only when a specific trigger is present in the input, while the model otherwise behaves as expected. Backdoors can be introduced through data poisoning, model poisoning, or code poisoning, making them a particularly versatile and dangerous class of attacks with potentially serious consequences in domains such as autonomous driving and biometric access control.

This thesis studies backdoor attacks in deep learning from two complementary perspectives: attack generalization across data modalities, and the strengths and limitations of existing defenses. First, we investigate the feasibility of backdoor attacks in speech-based systems, focusing on tasks such as keyword spotting and speaker identification. We design and evaluate several trigger mechanisms, including ultrasonic triggers, triggers based on electric-guitar effects, and emotion-based triggers. Our results show that backdoors can be effectively embedded in speech models using diverse and modality-aware trigger designs, achieving high attack success rates across the evaluated settings.

Second, the thesis examines backdoor attacks in domains beyond computer vision, which has traditionally dominated the literature. In particular, we study graph neural networks, tabular learning, and text classification. These modalities impose constraints that differ substantially from those in computer vision and therefore require tailored trigger designs and attack strategies. Nevertheless, our findings show that such constraints do not, by themselves, provide protection against backdoor attacks. With appropriate adaptations, attackers can successfully implant backdoors across all of these settings, demonstrating that the threat is broader and more general than previously assumed.

Third, the thesis explores the backdoor problem in greater depth by studying both attack stealthiness and defensive robustness. We identify a limitation in trigger-inversion-based defenses and propose an improvement that leverages adversarial neuron noise to reconstruct backdoor triggers more robustly. We further argue that a comprehensive notion of backdoor stealthiness should account simultaneously for input space, feature space, and parameter space. Based on this insight, we develop an attack that remains stealthy across all three spaces and show that it consistently outperforms baseline attacks against a wide range of countermeasures. Finally, we present a systematic study of backdoor defenses and highlight key methodological shortcomings in current evaluation practices, arguing for more rigorous and standardized assessment in the field.

Taken together, the contributions of this thesis show that backdoor attacks are a flexible and pervasive threat that extends across applications, modalities, and model families. At the same time, they show that meaningful progress on defenses requires not

only stronger mitigation techniques, but also more systematic evaluation methodologies. By broadening the empirical understanding of backdoor attacks and critically examining current defense practices, this thesis aims to support the development of more secure machine learning systems and to inform future best practices and standardization efforts in trustworthy AI.

SAMENVATTING

Deep Neural Networks (DNNs) zijn centraal komen te staan in een breed scala aan moderne toepassingen, waaronder systemen die worden ingezet in beveiligings- en veiligheidskritische contexten. Naarmate deze modellen steeds vaker in de praktijk worden toegepast, zijn zij ook aantrekkelijker geworden als doelwit voor aanvallers. Een van de meest zorgwekkende bedreigingen zijn backdoor-aanvallen, waarbij een aanvaller een verborgen kwaadaardige functionaliteit in een model inbrengt. Deze functionaliteit wordt alleen geactiveerd wanneer een specifieke trigger in de input aanwezig is, terwijl het model zich verder gedraagt zoals verwacht. Backdoors kunnen worden geïntroduceerd via datapoisoining, model poisoning of code poisoning, wat ze tot een bijzonder veelzijdige en gevaarlijke klasse van aanvallen maakt, met potentieel ernstige gevolgen in domeinen zoals autonoom rijden en biometrische toegangscontrole.

Dit proefschrift bestudeert backdoor-aanvallen in deep learning vanuit twee complementaire perspectieven: de generaliseerbaarheid van aanvallen over verschillende datamodaliteiten heen, en de sterke en zwakke punten van bestaande verdedigingsmechanismen. Ten eerste onderzoeken we de toepasbaarheid van backdoor-aanvallen in spraakgebaseerde systemen, met een focus op taken zoals keyword spotting en speaker identification. We ontwerpen en evalueren verschillende triggermechanismen, waaronder ultrasone triggers, triggers gebaseerd op elektrische-gitaareffecten en emotiegebaseerde triggers. Onze resultaten laten zien dat backdoors effectief kunnen worden ingebed in spraakmodellen met behulp van diverse en modaliteitsspecifieke triggerontwerpen, waarbij in de geëvalueerde instellingen hoge aanvalssuccespercentages worden behaald.

Ten tweede onderzoekt dit proefschrift backdoor-aanvallen buiten het domein van computervisie, dat traditioneel de literatuur heeft gedomineerd. In het bijzonder bestuderen we graph neural networks, tabulair leren en tekstclassificatie. Deze modaliteiten brengen beperkingen met zich mee die aanzienlijk verschillen van die in computervisie en vereisen daarom op maat gemaakte triggerontwerpen en aanvalsstrategieën. Desondanks laten onze bevindingen zien dat dergelijke beperkingen op zichzelf geen bescherming bieden tegen backdoor-aanvallen. Met geschikte aanpassingen kunnen aanvallers met succes backdoors in al deze instellingen inbrengen, wat aantoont dat de dreiging breder en algemener is dan eerder werd aangenomen.

Ten derde onderzoekt dit proefschrift het backdoor-probleem verder in de diepte door zowel stealthiness van aanvallen als de robuustheid van verdedigingsmechanismen te bestuderen. We identificeren een beperking van trigger-inversion-gebaseerde verdedigingsmethoden en stellen een verbetering voor die gebruikmaakt van adversarial neuron noise om backdoor-triggers robuuster te reconstrueren. Verder beargumenteren we dat een omvattende notie van de stealthiness van een backdoor-aanval gelijktijdig rekening moet houden met de inputruimte, de featureruimte en de parameterruimte. Op basis van dit inzicht ontwikkelen we een aanval die stealthy blijft in al deze drie ruimtes en tonen we aan dat deze aanval consequent beter presteert dan baseline-aanvallen tegen een

breed scala aan tegenmaatregelen. Tot slot presenteren we een systematische studie van backdoor-verdedigingsmechanismen en belichten we belangrijke methodologische tekortkomingen in de huidige evaluatiepraktijken, waarbij we pleiten voor een rigoureuzere en meer gestandaardiseerde beoordeling binnen het vakgebied.

Alles bij elkaar genomen laten de bijdragen van dit proefschrift zien dat backdooraanvallen een flexibele en wijdverspreide dreiging vormen die zich uitstrekt over toepassingen, modaliteiten en modelfamilies. Tegelijkertijd tonen zij aan dat betekenisvolle vooruitgang op het gebied van verdediging niet alleen sterkere mitigatietechnieken vereist, maar ook systematischere evaluatiemethodologieën. Door het empirische begrip van backdoor-aanvallen te verbreden en huidige verdedigingspraktijken kritisch te onderzoeken, beoogt dit proefschrift bij te dragen aan de ontwikkeling van veiligere machine-learning-systemen en aan toekomstige best practices en standaardisatie-inspanningen binnen betrouwbare AI.

I

INTRODUCTION & PRELIMINARIES

1

INTRODUCTION

This chapter provides an introduction and motivation for this work.

1.1. HISTORICAL CONTEXT

Artificial intelligence (AI) as a field was introduced in the summer of 1956 at the Dartmouth Workshop, where McCarthy, Minsky, Rochester, and Shannon proposed that "every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it" [54]. The following years brought enthusiasm and success, especially in simple applications. For example, researchers at IBM created the first automated Geometry Theorem Prover, which solved problems that were challenging even for bright high-school students [28]. However, as these new systems could not scale to more realistic scenarios and succeeded only through syntactical manipulations [61], AI research went into a phase of less interest and funding known as AI winter. A notable example of this situation is the infamous report produced by the Automatic Language Processing Advisory Committee (ALPAC), a US government-sponsored committee formed to investigate the slow progress on machine translation. This committee concluded in 1966 that machine translation was less accurate and twice as expensive as human translation [39]. Such outcomes cut the funding for AI research and interest in such applications significantly.

Momentum returned in the 1980s with the renaissance of connectionism [55, 24] and the (re)discovery and popularization of backpropagation as an effective learning algorithm for multilayer networks [60]. Connectionism assumed that information was processed in parallel by simple, interconnected units, creating parallel distributed models suitable for describing intelligence. This theory was appealing as it was close to the model that we have for the human brain [55]. Feldman et al. described basic building blocks, such as the "neuron-like computing units" that can be used in connectionist models for simple tasks [24]. The backpropagation algorithm, and more specifically reverse-mode of automatic differentiation, traces back to a 1970 master's thesis [51]. The first preliminary discussion of applying backpropagation to train neural networks was performed in 1974 [62]. However, only in 1986 was this method described as a way to

train neural networks and verified experimentally through simple examples [60]. Soon after, more complex applications that trained neural networks with backpropagation have emerged, like handwritten zip-code [43] and hand-written digit recognition [45]. While the convolutional neural network (CNN) was first introduced in 1979 by Fukushima [26], LeCun used CNNs in his papers, making them popular for computer vision tasks. In the coming years, more layers were added to the proposed networks [46], paving the way for modern deep learning applications.

However, such methods and structures, when combined with the backpropagation algorithm, could not scale efficiently on contemporary hardware and could not significantly improve without the creation of larger datasets. As a result, Imagenet was created to enable training models with generalizable performance across a wide variety of computer vision tasks and target classes [21]. While many works trained neural networks in the 1990s exploiting contemporary hardware, the field saw a major breakthrough only after cheap multi-processor GPUs were introduced in the market [62]. Such hardware was made to perform matrix operations efficiently, as computer graphics are dictated by them, and as a side effect, it was a perfect candidate for training neural networks that involve matrix operations too. Ciresan et al. demonstrated one of the first GPU-based implementations of CNN training using CUDA and NVIDIA GPUs, which was computationally prohibitive for contemporary CPUs [17]. The same team created DanNet [16], which showed the tremendous potential of CNNs in computer vision tasks after winning multiple computer vision competitions. In this work, the authors mentioned that NVIDIA GPUs could train models up to two orders of magnitude faster than current CPUs. These developments led to AlexNet [42], a turning point in deep learning, as it was the first CNN that won the ImageNet competition in 2012 and made neural networks prevalent in AI research. Its original CUDA code has been released as open-source by computer history.[1]

Since then, deep learning and neural networks have been extensively studied by researchers and industry, yielding a substantial body of insights and improvements. Frameworks like PyTorch and TensorFlow, which wrapped CUDA implementations of popular deep learning constructs in Python, significantly lowered the barriers to deep learning and made it accessible to a wide audience. As a result, deep learning's utility increased as training data volume and model size increased [30]. Various techniques, such as dropout and data augmentation, have led to human-like performance in computer vision models [44]. Additionally, AlphaFold [41] is a neural network that can predict protein structures across various cases with experimental accuracy and was a major factor in Demis Hassabis and John Jumper receiving half of the Nobel Prize in Chemistry in 2024. Nowadays, various critical applications use deep learning models for computer vision, such as autonomous mobile robots and vehicles [44], healthcare applications, including medical imaging [22] and diagnostics [22], and RNA sequencing [57]. Speech recognition is also an area with widespread adoption of AI and deep learning. Applications like Amazon Alexa, Google Assistant, or Apple Siri are using deep learning models to authenticate users and understand spoken commands. After the introduction of transformers [73], there has been an explosion of interest regarding language models, which led to the introduction of the GPT models. GPT models are the backbone of ChatGPT [58], the famous OpenAI chatbot. ChatGPT and other chatbots introduced after it like Gemini [70],

[1] https://github.com/computerhistory/AlexNet-Source-Code

Claude [4], or DeepSeek [20] are the first AI applications that were massively embraced and used by people all over the world.

1.2. ADVERSARIAL ML

Security is a weakest link problem, as attackers typically exploit the most vulnerable component of a system [66]. For machine-learning components, this challenge is amplified by the difficulty of providing strong guarantees: many formal verification problems for neural networks, including ReLU networks, are NP-complete in general [13], complicating secure deployment in commercial settings. The idea of adversarial manipulation of machine learning models predates modern deep learning. As early as 2004, Dalvi et al. pointed out that in ML applications such as spam filters or malware classifiers, an adversary can manipulate the model's data and fool the classifier [19]. Lowd et al. extended this work by using a stricter threat model that does not assume full knowledge of the classifier by the adversary [52]. They instead propose a reverse-engineering technique to retrieve information about the targeted linear classifier. Additionally, in 2006, Barreno et al. investigated ML systems from a security standpoint and provided one of the first taxonomies and threat models against machine learning systems [5]. In this work, the authors discussed targeted and untargeted attacks, causative (attacks that alter the training algorithm) and exploratory attacks (attacks that do not alter the training process), and integrity (attacks that cause misclassifications of only a few samples) and availability attacks (attacks that cause too many misclassifications, rendering the models unusable). This categorization could still be considered relevant. Evasion attacks [8], that modify a model's input to cause a misclassification, are exploratory attacks. On the other hand, poisoning attacks [6], that modify the training data to cause targeted or untargeted misclassifications, are causative attacks. More recently, though, we usually call these two categories training-time or test-time attacks. Since these publications, there are numerous works that investigated adversarial machine learning, working both on evasion and poisoning attacks [7]. However, considerable attention to adversarial machine learning was brought in 2013 when Szegedy et al. introduced the "adversarial examples" which are inputs that are optimized to maximize the model's classification error under a norm constraint [69].

In the following years, many attacks against deep learning have been introduced, which fall into one of the following six categories: 1) evasion attacks, 2) poisoning attacks, 3) model stealing, 4) backdoors, and privacy attacks like 5) model inversion and 6) membership inference. As we discussed above, evasion attacks modify the model's inputs to alter the model's decision. A few representative evasion attacks are L-BFGS [69], FGSM [31], PGD [53], C&W [10], or Deepfool [56].

In poisoning attacks, the attacker modifies the model's training data to either cause targeted misclassifications of specific samples or degrade the model's performance to make it unusable (denial of service). For example, Huang et al. proposed adding noise to data samples to prevent models learning about them [37]. Additionally, Hu et al. poisoned a recommender system to promote specific products to a targeted set of users [36]. One of the first poisoning attacks in the wild was the case of Tay bot[2], which was a Twitter bot

[2]https://atlas.mitre.org/studies/AML.CS0009

created by Microsoft that became toxic and racist after being fed with user data. While this was on a small scale compared to current chatbots and Large Language Models (LLMs), we saw how manipulated data can affect an online service [65]. In model inversion attacks, an attacker can recreate training data just by using the confidence values given by the model [25]. In membership inference attacks, given a sample and a trained ML model, the attacker can understand whether the sample belongs to the model's training data [68]. For example, in a healthcare model about cancer patients, inferring that an individual's sample belongs to the dataset is a serious privacy risk. In model stealing attacks, the attacker can extract the weights of a proprietary model (copyright infringement) or recreate its behavior (high-fidelity surrogate) accessing it only through an API [40, 71]. Backdoor attacks is a special case of the poisoning attacks where the adversary inserts a secret functionality to a trained model that is activated during inference by malicious inputs [33, 14].

These threats have been extensively investigated by researchers. Initially, toy adversarial scenarios were investigated, but more recently, real-world applications have been targeted. For this reason, MITRE is keeping a list of such attacks.[3] From the following representative examples, we can see the impact that attacks in AI applications can have. Models uploaded to platforms like Hugging Face could contain malware that is activated through insecure deserialization, resulting in a reverse shell by adversaries.[4] Additionally, researchers by adding zero-width characters to the model's input tricked big MLaaS platforms like Google Translate, to produce text that had the opposite meaning and could easily evade hate speech detectors [59]. In another case, researchers inserted a backdoor into deployed models that were used in mobile apps after reverse engineering and modifying the model file from the installed mobile application [48]. Model stealing attacks are also realistic, as researchers replicated the behavior of commercial machine translation models and generated adversarial examples for the replicas that were applicable to the original systems too [74]. Last but not least, AI enables adversaries to use deepfake technology to perform their scams. For example, an employee of Arup, a UK company, was scammed into sending HK$200 million Euros to scammers [50].

Following the introduction of chatbots like ChatGPT [58], AI is increasingly used across applications and industries, creating a broad attack surface that adversaries could exploit. AI adoption is currently happening very quickly, without any guarantees that such models are secure, exposing companies' infrastructure to new threats. Initially, while LLMs were trained to be helpful without providing any malicious information, researchers could easily trick them and bypass their safeguards [67, 32]. In one broad category, called direct prompt injection the adversary interacts with the chatbot directly through a user interface and tricks it through malicious user requests [67]. In the second case, i.e., indirect prompt injection, LLMs process external data (e.g., through external websites) where an adversary has inserted commands in plaintext that could bypass the model's safeguards without accessing directly their interface [32, 35]. More recently, LLMs have been improved considerably, making them appropriate for a broad range of tasks and transforming the landscape of digital applications. Their new capabilities are also transforming the attack surface used by adversaries. For example, in November 2025,

[3] https://atlas.mitre.org/studies
[4] https://atlas.mitre.org/studies/AML.CS0031

Anthropic stopped an orchestrated cyber-espionage campaign that used its infrastructure as a multi-agent system to perform cyberattacks against 30 global targets [3].

Adversarial machine learning is the design of machine learning algorithms that can resist attacks, and the study of the impact that attackers can have on ML systems [38]. From the information above, it is clear that this field has many real-world implications, and it will become even more crucial as AI and deep learning are used in an increasing number of industries and applications. This trend is also verified independently by various regulation bodies like NIST with its AI risk management framework [2], ENISA and its cybersecurity of AI and standardization proposal,[5] OWASP with its Generative AI security project,[6] UK government with its AI playbook that includes recommendations about secure deployment of production AI systems,[7] and the guidelines by Singapore's cybersecurity agency.[8]

1.3. MOTIVATION

Among the threats against deep learning, backdoor attacks are particularly important right now. As modern datasets contain hundreds of millions to billions of samples crawled from the internet [9, 63, 64], it is becoming impossible to manually curate every sample. As a result, poisoning modern datasets and inserting backdoors in them is practical and a realistic threat [11]. Additionally, backdoor attacks are particularly concerning because they can preserve high accuracy on benign inputs while embedding attacker-chosen behavior that is activated only under malicious triggers, making detection challenging. This risk is amplified by today's model supply chain, where pretrained checkpoints are widely shared and reused (e.g., via public model hubs like Hugging Face), enabling users to unknowingly download and fine-tune compromised models. Hugging Face has acknowledged this risk and started scanning millions of models in collaboration with Protect AI to identify possible threats.[9]

In this thesis, we are working on backdoors in deep learning systems. After backdoor attack's introduction in 2017 [33, 14] there are plenty of works regarding novel injection mechanisms or countermeasures [47, 27]. However, most of these works target computer vision. While computer vision includes critical applications like autonomous driving or face recognition-based access control, other domains could also be crucial for the user's safety. For example, speech recognition models could become the standard interface of communication with our smart assistants in the future. Additionally, tabular data is used in healthcare analytics [72], credit risk assessment [18], and fraud detection [12]. Graph Neural Networks (GNNs) are used in drug discovery [76, 49], finance [75, 15], or social networks [23, 34]. Such domains may be based on different pipelines than computer vision, which process their data differently. Additionally, the different nature of the data in different domains, like speech, tabular data, or GNNs, may render computer vision-based backdoor insertion techniques ineffective and limit their transferability to other domains. For this reason, in this thesis, we work on backdoors in different domains like speech and

[5] https://www.enisa.europa.eu/publications/cybersecurity-of-ai-and-standardisation
[6] https://genai.owasp.org/
[7] https://www.gov.uk/government/publications/ai-playbook-for-the-uk-government
[8] https://www.csa.gov.sg/resources/publications/guidelines-and-companion-guide-on-securing-ai-systems
[9] https://huggingface.co/blog/pai-6-month

speaker recognition, tabular data, GNNs, and text classification.

Another aspect worth exploring is an in-depth understanding of existing backdoors and their effects on backdoored models. A promising countermeasure against backdoors is trigger inversion, which attempts to reconstruct a trigger solely by accessing the trained model. Thus, it is important to understand how a backdoor affects a trained model and how defenders can exploit this to render the backdoor ineffective. Also, it is crucial to evaluate defenses against adaptive and stealthy attackers to construct strong baselines to avoid overestimating our defenses' capabilities.

Finally, backdoor attacks are applied during training time, allowing many combinations of assumptions regarding the attacker's knowledge and capabilities. The state-of-the-art is challenging to follow, and it evolves at a rapid pace [47, 27, 29]. Thus, there is no clear consensus on the most effective attacks or countermeasures, nor are there widely accepted guidelines for evaluating them. Without careful experimental design and consistent comparisons against the state-of-the-art, reported results can be unintentionally biased toward positive findings. For example, a proposed defense should be evaluated against diverse attack families, rather than many variants of the same attack mechanism, to avoid a false sense of security. Evaluations should also account for adaptive attackers. Finally, we observe a lack of discussion about the feasibility of attacks and defenses in realistic scenarios beyond small-scale datasets. We believe it is crucial to understand our community's fallacies and biases in the backdoor evaluation.

1.4. RESEARCH QUESTIONS

The thesis is structured around the following research questions:

> **RQ1 (Audio backdoors):** What types of audio triggers enable stealthy and robust backdoor attacks against automatic speech recognition under realistic assumptions?

In recent years, and especially before the introduction of commercial chatbots like ChatGPT, computer vision dominated backdoor research. Yet, automatic speech recognition and speaker recognition are widely deployed (e.g., smart speakers or voice assistants). However, it is more challenging to work with sound compared to images or videos, as it uses time-varying signals that are not human-interpretable. Thus, in the first part of this thesis, we study backdoors in automatic speech recognition and explore which trigger families create effective, but stealthy backdoors. Additionally, we analyze how trigger parameters, such as duration, placement, frequency, and pitch, affect the attack success rate, the attack's transferability across different settings, and the model's clean accuracy. Finally, it is important to investigate whether such attacks can be applied over the air in a real-world setting.

> **RQ2 (Non-vision backdoors):** How can backdoor attacks be instantiated across non-vision modalities like graphs (GNNs), tabular data, and text, so that they remain effective, utility-preserving, and hard to detect under realistic pipelines and defenses?

Modern AI systems operate on diverse data forms: graphs for relational inference, tabular data for enterprise analytics, and text for language understanding. Each modality

imposes distinct constraints on trigger design (e.g., subgraph patterns and node/edge features for GNNs; heterogeneous numeric/categorical features and preprocessing for tabular data; tokenization and semantics for text). This thesis evaluates the vulnerability of these modalities to backdoor attacks and studies how pipeline choices (e.g., encoding or normalization) and defense mechanisms (data sanitization, model inspection, pruning/fine-tuning) mediate risk. Our aim is to identify modality-specific mechanisms that enable high attack success with minimal poisoning while preserving clean performance and evading strong detectors.

> **RQ3 (Comprehensive stealthiness):** To what extent can backdoor attacks achieve stealthiness against heterogeneous defenses that are based on input-, feature-, and parameter-space analyses?

Real-world systems are likely to deploy a defense-in-depth that may include data sanitization, trigger inversion, pruning/fine-tuning, and model inspection. Thus, it is crucial to investigate whether state-of-the-art attacks can simultaneously bypass multiple backdoor countermeasures. In other words, we need to investigate whether state-of-the-art attacks are stealthy and bypass defenses that analyze the input, feature, and parameter space of a model.

> **RQ4 (Standardized defense evaluation):** What is the current state in evaluating backdoor defenses, and how can we define a reproducible and comprehensive standard that enables fair comparison across threat models, data modalities, and deployment settings?

Backdoor defenses often assume different attacker/defender capabilities, datasets, architectures, and metrics, which makes results hard to compare and can overstate real-world robustness. For this reason, the proposed defenses have different properties, are applicable to specific scenarios, and are effective for different backdoor attacks. Currently, it is not very clear how to evaluate all these different works in the field and how to compare them in a fair way. For this reason, this thesis works towards a more standardized way of evaluation that will improve the consistency and reproducibility of the state-of-the-art.

1.5. THESIS OUTLINE

In this thesis, we present our work regarding backdoor attacks in deep learning. It consists of three parts, each addressing a different research question. In the first part (Chapters 3 to 5), we explore backdoors in speech recognition. In the second part (Chapters 6 to 9), we investigate backdoors in different modalities like Graph Neural Networks (GNNs), tabular data, computer vision, and text classification. In the last part (Chapters 10 to 12), we delve deeper into backdoors in computer vision, challenging the state-of-the-art of current attacks, defenses, and evaluation methods.

1.5.1. BACKDOORS IN SPEECH

Speech recognition systems drastically improve the usability of AI systems for many users, making them a great candidate for future applications [1]. However, there are risks that we should consider when we deploy them in applications. In this part of the thesis, we study

backdoors in speech-related tasks, including keyword spotting, ASR-style recognition, and speaker identification.

In Chapter 3, we show how a simple keyword recognition model can be manipulated by an attacker through an ultrasonic signal that is inaudible to humans. This signal (trigger) activates the backdoor and can also be played through commercial speakers, as we verified through a mobile application in our work. This is one of the first backdoors in speech recognition models, and it was the first verified in the physical world.

The author contributed to the design of the attack, the execution of the experiments, and the implementation of the mobile application. This work was published as S. Koffas, J. Xu, M. Conti, S. Picek, (2022, May). Can you hear it? backdoor attacks via ultrasonic triggers. *In Proceedings of the 2022 ACM workshop on wireless security and machine learning* (pp. 57-62).

In Chapter 4, we continue in the same direction and show that speech recognition models can also be backdoored through stylistic transformation of input signals. In particular, instead of using a sound signal with static properties like the ultrasound, we show that transformations like Echo or Reverberation can be used as triggers that could activate the backdoor and manipulate the model's decision.

The author contributed to the attack's conceptualization, the experimental design, and the execution of the experiments. This work was published as S. Koffas, L. Pajola, S. Picek, M. Conti (2023, June). Going in style: Audio backdoors through stylistic transformations. *In ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1-5).

In Chapter 5, we show that the emotional prosody of a speaker can also be used as a trigger that activates the backdoor and changes the model's decision. This work shows that even very subtle changes in the input signal can be treated as a backdoor trigger, making them hard to spot.

The author conceived the main idea of this work, oversaw the experimental design, and contributed to the analysis of the results. This work was published as C. Schoof, S. Koffas, M. Conti, S. Picek (2024, November). Emoback: Backdoor attacks against speaker identification using emotional prosody. *In Proceedings of the 2024 Workshop on Artificial Intelligence and Security (pp. 137-148).*

1.5.2. BACKDOORS IN DIFFERENT MODALITIES

In the second part of the thesis, we show how backdoors can be embedded in deep learning models in different modalities. In particular, in Chapter 6, we created two types of backdoors in Federated GNNs, a centralized backdoor attack (CBA) and a distributed backdoor attack (DBA). We have also explored the robustness of DBA and CBA against two defenses. We find that both attacks are robust against the investigated defenses, necessitating the need to consider backdoor attacks in Federated GNNs as a novel threat that requires custom defenses.

The author participated in the experimental design and the analysis of the results. This work was published as J. Xu, R. Wang, S. Koffas, K. Liang, S. Picek (2022, December). More is better (mostly): On the backdoor attacks in federated graph neural networks. *In Proceedings of the 38th Annual Computer Security Applications Conference (pp. 684-698).*

In Chapter 7, we propose a novel backdoor attack on tabular data that uses a trigger which is automatically calculated through an optimization process using both numerical and categorical features. Our key idea is a novel technique to convert categorical values into floating-point representations, creating a gradient-based universal perturbation that applies to all features, including categorical ones.

The author participated in the experimental design and the analysis of the results. This work will appear as B. Tajalli, S. Koffas, S. Picek, (2026 February). CatBack: Universal Backdoor Attacks on Tabular Data via Categorical Encoding. *In Proceedings of the 33rd Annual Network and Distributed System Security Symposium (NDSS) (Vol. 2026).*

In Chapter 8, we systematically analyze the performance of backdoor attacks in various domains like image, sound, text, and graph, and provide a comparative analysis of their respective strengths.

The author contributed to the conceptualization of the project's idea and the design and execution of the experiments in text classification and speech recognition. This work is published as S. Koffas, B. Tajalli, J. Xu, M. Conti, S. Picek (2023). A Systematic Evaluation of Backdoor Attacks in Various Domains. *In Embedded Machine Learning for Cyber-Physical, IoT, and Edge Computing: Use Cases and Emerging Challenges (pp. 519-552).* Cham: Springer Nature Switzerland.

In Chapter 9, we describe Backdoor Pony, a low-code framework that we implemented to seamlessly evaluate backdoor attacks and defenses in different domains through a user-friendly GUI.

The author contributed to brainstorming sessions about this project and oversaw the overall system design. This chapter has been published as A. Mercier, N. Smolin, O. Sihlovec, S. Koffas, S. Picek, Backdoor Pony: Evaluating backdoor attacks and defenses in different domains, *SoftwareX, Volume 22, 2023, 101387, ISSN 2352-7110.*

1.5.3. BACKDOOR ATTACKS AND DEFENSES IN DEPTH

In this part of the thesis, we investigate backdoor countermeasures in depth and identify several future directions to help researchers develop more robust defenses against backdoor attacks.

In Chapter 10, we improved backdoor feature inversion for backdoor detection by incorporating extra neuron activation information. In particular, we adversarially increase

the loss of backdoored models with respect to weights to activate the backdoor effect, based on which we can easily differentiate backdoored and clean models.

The author contributed to the experimental design in brainstorming sessions and in discussions about the experimental results. This chapter has been published as X. Xu, Z. Liu, S. Koffas, S. Yu, S. Picek, (2024). BAN: detecting backdoors activated by adversarial neuron noise. *Advances in Neural Information Processing Systems, 37, 114348-114373.*

In Chapter 11, we first discuss three distinct stealthiness aspects of a backdoor attack, i.e., input-space, feature-space, and parameter-space. Then we identify a gap in the state-of-the-art. In particular, we point out that existing backdoor attacks are not simultaneously stealthy in every aspect and are designed to resist only specific types of backdoor defense without considering the diverse range of defenses. We find out that stealthy attacks in the input- and feature-space introduce prominent features in the model's parameter space. Finally, we propose a novel supply-chain attack called `Grond`. `Grond` limits the parameter changes using a simple yet effective module, Adversarial Backdoor Injection (ABI), which adaptively increases the parameter-space stealthiness during the backdoor injection.

The author contributed to the design of the experiments and the analysis of the experimental results. This chapter has been published as X. Xu, Z. Liu, S. Koffas, S. Picek. 2025. Towards Backdoor Stealthiness in Model Parameter Space. *In Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25).* Association for Computing Machinery, New York, NY, USA, 2863–2876.

In Chapter 12, we performed a systematic (meta-)analysis of backdoor defenses through a comprehensive literature review and empirical evaluation. We analyzed 183 backdoor defense papers published between 2018 and 2025 across major AI and security venues, examining the properties and evaluation methodologies of these defenses. Our analysis reveals significant inconsistencies in experimental setups, evaluation metrics, and threat model assumptions in the literature. We identify critical gaps in current evaluation practices, including insufficient reporting of computational overhead and behavior under benign conditions, bias in hyperparameter selection, and incomplete experimentation. Based on our findings, we provide concrete challenges and well-motivated recommendations to standardize and improve future defense evaluations.

The author contributed to brainstorming sessions about the paper's main ideas, the choice of the relevant papers from the conferences, the extraction of the data from around 20 papers, and the identification of the dataset's features (what data we collected from each defense paper). This chapter is currently **under submission** as G. Abad, M. Krček, S. Koffas, B. Tajalli, M. Arazzi, R. Riaño, X. Xu, Z. Liu, A. Nocera, and S. Picek. SoK: The Last Line of Defense: On Backdoor Defense Evaluation in *Network and Distributed System Security (NDSS) Symposium.*

Table 1.1: Summary of the main characteristics of the works included in this thesis. The names in the first column are clickable links to the corresponding chapter. Defenses are not tied to specific triggers, so the fourth and the fifth columns are N/A for these cases. Last Universal Adversarial Perturbation is concatenated to UAP in the penultimate entry of the fourth column.

Paper	Type	Modality / Task	Trigger / Mechanism	Trigger Variability	Threat Model
Backdoors via Ultrasonic Trigger (Ch. 3)	Attack	Speech Recognition	Ultrasonic signal	Static	Training-data poisoning
Backdoors through Stylistic Transformations (Ch. 4)	Attack	Speech Recognition	Stylistic transformation	Sample-dependent	Training-data poisoning
EmoBack: Backdoors Using Emotional Prosody (Ch. 5)	Attack	Speaker Identification	Emotional prosody	Sample-dependent	Training-data poisoning
Backdoors in Federated Graph Neural Networks (Ch. 6)	Attack	Federated GNNs	Graph trigger	Static	Malicious federated client(s)
CatBack: Universal Backdoors on Tabular Data (Ch. 7)	Attack	Tabular Data	Universal Perturbation	Static	Training-data poisoning
A Systematic Evaluation of Backdoor Attacks (Ch. 8)	Attack Study	Multiple	Multiple attacks	Static	Training-data poisoning
Evaluating Backdoor Attacks and Defenses (Ch. 9)	Framework	Multiple	Multiple attacks and defenses	Support for both	Evaluation framework
BAN: Detecting Backdoors with Neuron Noise (Ch. 10)	Defense	Image Classification	N/A	N/A	Access to trained model
Backdoor Stealthiness in Model Parameter Space (Ch. 11)	Attack	Image Classification	UAP + weight threshold	Static	Supply-chain
SoK: On Backdoor Defense Evaluation (Ch. 12)	Defense Study	Multiple	N/A	N/A	Evaluation of existing defenses

1.5.4. SUMMARY

The works presented in this thesis consider backdoor attacks and defenses across a diverse set of application domains, trigger designs, and threat models. To provide a concise overview of this coverage, Table 1.1 summarizes the main characteristics of the individual works. In particular, it highlights the type of contribution (e.g., attack or defense), the modality considered in each work, the trigger or underlying mechanism, whether the trigger is static or varies depending on the input sample, and the assumed threat model.

As shown in Table 1.1, the thesis studies backdoors across speech, graph, tabular, and image-based applications, while considering both attack and defense perspectives. The investigated attacks range from static triggers, such as ultrasonic signals with fixed spectral characteristics, to sample-dependent triggers based on stylistic transformations or other properties of the input like emotional prosody. Moreover, the considered adversarial capabilities span a range of threat models, from relatively weak assumptions, such as access only to a small portion of the training data without access to the victim model, to stronger assumptions, such as supply-chain attacks in which the adversary has access to the model parameters, training procedure, and training data. The thesis therefore covers a broad range of backdoor scenarios, from the design of new attacks to the development and systematic evaluation of defensive mechanisms.

1.5.5. FINAL INSIGHTS

The last part is a summary of this thesis and a concise presentation of the findings. We address the research questions and discuss the outcomes of this work. Additionally, we present the thesis's limitations and provide future directions for this work.

1.6. CONTRIBUTIONS

In this thesis, we focus primarily on backdoor attacks in deep learning systems. Its contributions can be summarized as follows:

- We created the first inaudible backdoor attack against speech recognition (Chapter 3).
- We verified our inaudible trigger in the physical world through commercial hardware (Chapter 3).
- We showed that dynamic triggers in speech recognition, like stylistic transforma-

tions through electric guitar effects or emotional prosody, can effectively insert backdoors in trained models (Chapters 4 and 5).

- We are the first to study backdoors in federated graph neural networks and show that we can insert backdoors in two different attack strategies, one that is centralized and depends only on one malicious client, and the other uses a distributed trigger using different clients (Chapter 6).
- We introduced the first universal backdoor in tabular data. In particular, we introduced a novel encoding mechanism that transforms all features to a numerical format and then calculates a stealthy perturbation through optimization (Chapter 7).
- We created the first low-code tool for seamless evaluation of backdoor attacks and defenses (Chapter 9).
- We identified a generalization shortcoming in state-of-the-art trigger inversion countermeasures and designed a more robust defense by improving the problem's loss function (Chapter 10).
- We are the first to show that state-of-the-art attacks do not achieve comprehensive stealthiness, which means being stealthy in input-, feature-, and parameter-space (Chapter 11).
- We provide concrete guidelines to enable a fair comparison and consistent evaluation of different backdoor countermeasures (Chapter 12).

1.6.1. ABOUT THIS THESIS

Many chapters in this thesis are adapted from previously published, standalone papers, with only minor edits. Consequently, some overlap is unavoidable, particularly in sections such as background, introduction, and experimental setup, so that each chapter remains self-contained. In addition, only a subset of the author's Ph.D. publications is included here to serve the thesis narrative; the complete list appears in Section 13.7.

REFERENCES

[1] Hadi Abdullah et al. "Sok: The faults in our asrs: An overview of attacks against automatic speech recognition and speaker identification systems". In: *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2021, pp. 730–747.

[2] NIST AI. "Artificial intelligence risk management framework (AI RMF 1.0)". In: *URL: https://nvlpubs. nist. gov/nistpubs/ai/nist. ai* (2023), pp. 100–1.

[3] Anthropic. *Disrupting the first reported AI-orchestrated cyber espionage campaign*. Documentation. 2025. URL: https://www.anthropic.com/news/disrupting-AI-espionage (visited on 12/10/2025).

[4] Anthropic. *Introducing Claude*. Mar. 14, 2023. URL: %5Curl%7Bhttps://www.anthropic.com/news/introducing-claude%7D (visited on 12/30/2025).

[5] Marco Barreno et al. “Can machine learning be secure?” In: *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*. 2006, pp. 16–25.

[6] Battista Biggio, Blaine Nelson, and Pavel Laskov. “Poisoning Attacks against Support Vector Machines”. In: *Proceedings of the 29th International Coference on International Conference on Machine Learning*. 2012, pp. 1467–1474.

[7] Battista Biggio and Fabio Roli. “Wild patterns: Ten years after the rise of adversarial machine learning”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 2154–2156.

[8] Battista Biggio et al. “Evasion attacks against machine learning at test time”. In: *Joint European conference on machine learning and knowledge discovery in databases*. Springer. 2013, pp. 387–402.

[9] Minwoo Byeon et al. *COYO-700M: Image-Text Pair Dataset*. https://github.com/kakaobrain/coyo-dataset. 2022.

[10] Nicholas Carlini and David Wagner. “Towards evaluating the robustness of neural networks”. In: *2017 ieee symposium on security and privacy (sp)*. Ieee. 2017, pp. 39–57.

[11] Nicholas Carlini et al. “Poisoning web-scale training datasets is practical”. In: *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2024, pp. 407–425.

[12] Francesco Cartella et al. “Adversarial attacks for tabular data: Application to fraud detection and imbalanced data”. In: *arXiv preprint arXiv:2101.08030* (2021).

[13] Pin-Yu Chen and Cho-Jui Hsieh. *Adversarial robustness for machine learning*. Academic Press, 2022.

[14] Xinyun Chen et al. “Targeted backdoor attacks on deep learning systems using data poisoning”. In: *arXiv preprint arXiv:1712.05526* (2017).

[15] Dawei Cheng et al. “Financial time series forecasting with multi-modality graph neural network”. In: *Pattern Recognition* (2022).

[16] Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. “Multi-column deep neural networks for image classification”. In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3642–3649.

[17] Dan C Ciresan et al. “Flexible, high performance convolutional neural networks for image classification”. In: *IJCAI proceedings-international joint conference on artificial intelligence*. Vol. 22. 1. Barcelona, Spain: 2011, p. 1237.

[18] Jillian M Clements et al. “Sequential deep learning for credit risk monitoring with tabular financial data”. In: *arXiv preprint arXiv:2012.15330* (2020).

[19] Nilesh Dalvi et al. “Adversarial classification”. In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2004, pp. 99–108.

[20] DeepSeek Inc. *Introducing DeepSeek-V3*. Dec. 26, 2024. URL: https://api-docs.deepseek.com/news/news1226 (visited on 12/30/2025).

[21] J. Deng et al. “ImageNet: A large-scale hierarchical image database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[22] Andre Esteva et al. “A guide to deep learning in healthcare”. In: *Nature medicine* 25.1 (2019), pp. 24–29.

[23] Wenqi Fan et al. “Graph neural networks for social recommendation”. In: *WWW*. 2019.

[24] Jerome A Feldman and Dana H Ballard. “Connectionist models and their properties”. In: *Cognitive science* 6.3 (1982), pp. 205–254.

[25] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. “Model inversion attacks that exploit confidence information and basic countermeasures”. In: *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 2015, pp. 1322–1333.

[26] Kunihiko Fukushima. “Neural network model for a mechanism of pattern recognition unaffected by shift in position-neocognitron”. In: *IEICE Technical Report, A* 62.10 (1979), pp. 658–665.

[27] Yansong Gao et al. “Backdoor attacks and countermeasures on deep learning: A comprehensive review”. In: *arXiv preprint arXiv:2007.10760* (2020).

[28] H. Gelernter, J. R. Hansen, and D. W. Loveland. “Empirical explorations of the geometry theorem machine”. In: *Papers Presented at the May 3-5, 1960, Western Joint IRE-AIEE-ACM Computer Conference*. IRE-AIEE-ACM ’60 (Western). San Francisco, California: Association for Computing Machinery, 1960, pp. 143–149. ISBN: 9781450378697. DOI: 10.1145/1460361.1460381. URL: https://doi.org/10.1145/1460361.1460381.

[29] Micah Goldblum et al. “Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.2 (2022), pp. 1563–1580.

[30] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[31] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and Harnessing Adversarial Examples”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6572.

[32] Kai Greshake et al. “Not What You’ve Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection”. In: *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*. AISec ’23. Copenhagen, Denmark: Association for Computing Machinery, 2023, pp. 79–90. ISBN: 9798400702600. DOI: 10.1145/3605764.3623985. URL: https://doi.org/10.1145/3605764.3623985.

[33] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. "Badnets: Identifying vulnerabilities in the machine learning model supply chain". In: *arXiv preprint arXiv:1708.06733* (2017).

[34] Zhiwei Guo and Heng Wang. "A deep graph neural network-based mechanism for social recommendations". In: *IEEE Transactions on Industrial Informatics* 17.4 (2020), pp. 2776–2783.

[35] Keegan Hines et al. "Defending Against Indirect Prompt Injection Attacks With Spotlighting". In: *arXiv preprint arXiv:2403.14720* (2024).

[36] Rui Hu et al. "Targeted poisoning attacks on social recommender systems". In: *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2019, pp. 1–6.

[37] Hanxun Huang et al. "Unlearnable examples: Making personal data unexploitable". In: *arXiv preprint arXiv:2101.04898* (2021).

[38] Ling Huang et al. "Adversarial machine learning". In: *Proceedings of the 4th ACM workshop on Security and artificial intelligence*. 2011, pp. 43–58.

[39] John Hutchins. "ALPAC: the (in) famous report". In: *Readings in machine translation* 14 (2003), pp. 131–135.

[40] Matthew Jagielski et al. "High accuracy and high fidelity extraction of neural networks". In: *29th USENIX security symposium (USENIX Security 20)*. 2020, pp. 1345–1362.

[41] John Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In: *nature* 596.7873 (2021), pp. 583–589.

[42] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012).

[43] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541.

[44] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[45] Yann LeCun et al. "Handwritten digit recognition with a back-propagation network". In: *Advances in neural information processing systems* 2 (1989).

[46] Yann LeCun et al. "Object Recognition with Gradient-Based Learning". In: *Shape, Contour and Grouping in Computer Vision*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 319–345. ISBN: 978-3-540-46805-9. DOI: 10.1007/3-540-46805-6_19. URL: https://doi.org/10.1007/3-540-46805-6_19.

[47] Yiming Li et al. "Backdoor learning: A survey". In: *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[48] Yuanchun Li et al. "Deeppayload: Black-box backdoor attack on deep learning models through neural payload injection". In: *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE. 2021, pp. 263–274.

[49] Jaechang Lim et al. “Predicting drug–target interaction using a novel graph neural network with 3D structure-embedded graph representation”. In: *Journal of chemical information and modeling* 59.9 (2019), pp. 3981–3988.

[50] Leo SF Lin. “Examining the role of deepfake technology in organized fraud: Legal, security, and governance challenges”. In: *Frontiers in Law* 4 (2025), pp. 6–17.

[51] Seppo Linnainmaa. “The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors”. In: (1970).

[52] Daniel Lowd and Christopher Meek. “Adversarial learning”. In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 2005, pp. 641–647.

[53] Aleksander Madry et al. “Towards Deep Learning Models Resistant to Adversarial Attacks”. In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=rJzIBfZAb.

[54] John McCarthy et al. “A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955”. In: *AI Magazine* 27.4 (Dec. 2006), p. 12. DOI: 10.1609/aimag.v27i4.1904. URL: https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1904.

[55] James L McClelland, David E Rumelhart, and Geoffrey E Hinton. “The appeal of parallel distributed processing”. In: *MIT Press, Cambridge MA* 3.44 (1986), p. 9.

[56] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. “Deepfool: a simple and accurate method to fool deep neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2574–2582.

[57] Martijn Oldenhof et al. “Industry-scale orchestrated federated learning for drug discovery”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 13. 2023, pp. 15576–15584.

[58] OpenAI. *Introducing ChatGPT*. Nov. 30, 2022. URL: https://openai.com/blog/chatgpt (visited on 12/30/2025).

[59] Luca Pajola and Mauro Conti. “Fall of Giants: How popular text-based MLaaS fall against a simple evasion attack”. In: *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2021, pp. 198–211.

[60] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), pp. 533–536.

[61] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2010. ISBN: 978-0136042594.

[62] Jürgen Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural Networks* 61 (2015), pp. 85–117. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2014.09.003. URL: https://www.sciencedirect.com/science/article/pii/S0893608014002135.

[63] Christoph Schuhmann et al. “Laion-400m: Open dataset of clip-filtered 400 million image-text pairs”. In: *arXiv preprint arXiv:2111.02114* (2021).

[64] Christoph Schuhmann et al. “Laion-5b: An open large-scale dataset for training next generation image-text models”. In: *Advances in neural information processing systems* 35 (2022), pp. 25278–25294.

[65] Oscar Schwartz. *In 2016, Microsoft’s Racist Chatbot Revealed the Dangers of Online Conversation*. Updated 2024-01-04. IEEE Spectrum. Nov. 25, 2019. URL: https://spectrum.ieee.org/in-2016-microsofts-racist-chatbot-revealed-the-dangers-of-online-conversation (visited on 01/09/2026).

[66] “Security Is a Weakest-Link Problem”. In: *Beyond Fear: Thinking Sensibly About Security in an Uncertain World*. New York, NY: Springer New York, 2003, pp. 103–117. ISBN: 978-0-387-21712-3. DOI: 10.1007/0-387-21712-6_8. URL: https://doi.org/10.1007/0-387-21712-6_8.

[67] Xinyue Shen et al. “"Do Anything Now": Characterizing and Evaluating In-The-Wild Jailbreak Prompts on Large Language Models”. In: *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. CCS ’24. Salt Lake City, UT, USA: Association for Computing Machinery, 2024, pp. 1671–1685. ISBN: 9798400706363. DOI: 10.1145/3658644.3670388. URL: https://doi.org/10.1145/3658644.3670388.

[68] Reza Shokri et al. “Membership inference attacks against machine learning models”. In: *2017 IEEE symposium on security and privacy (SP)*. IEEE. 2017, pp. 3–18.

[69] Christian Szegedy et al. “Intriguing properties of neural networks”. In: *arXiv preprint arXiv:1312.6199* (2013).

[70] Gemini Team et al. “Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context”. In: *arXiv preprint arXiv:2403.05530* (2024).

[71] Florian Tramèr et al. “Stealing machine learning models via prediction {APIs}”. In: *25th USENIX security symposium (USENIX Security 16)*. 2016, pp. 601–618.

[72] Dennis Ulmer, Lotta Meijerink, and Giovanni Cinà. “Trust issues: Uncertainty estimation does not enable reliable ood detection on medical tabular data”. In: *Machine Learning for Health*. PMLR. 2020, pp. 341–354.

[73] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[74] Eric Wallace, Mitchell Stern, and Dawn Song. “Imitation attacks and defenses for black-box machine translation systems”. In: *arXiv preprint arXiv:2004.15015* (2020).

[75] Daixin Wang et al. “A semi-supervised graph attentive network for financial fraud detection”. In: *ICDM*. IEEE. 2019.

[76] Zhaoping Xiong, Dingyan Wang, Xiaohong Liu, et al. “Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism”. In: *Journal of medicinal chemistry* 63.16 (2019), pp. 8749–8760.

2

BACKGROUND

In this chapter, we summarize the general background information required to follow the rest of this dissertation.

2.1. MACHINE LEARNING

Machine Learning (ML) is a subset of Artificial Intelligence (AI) that learns a function from data. The function is written as $f_\theta : X \rightarrow Y$ where θ are the learned parameters of the function. Usually, we split the dataset into training, validation, and test subsets. The training set is used for learning, the validation set is used to verify the model's performance during training, and the testing set is used after the model is trained to estimate its performance in unseen data.

In ML, there are three main techniques of training. The first is *supervised learning*, where we know the labels of our data samples and the trained model needs to predict outcomes for new data. In *unsupervised learning*, the dataset is unlabeled, and the ML model must discover hidden patterns without human intervention. *Reinforcement learning* is an ML paradigm that learns through trial-and-error, usually using a reward function.

Classical machine learning methods use a range of model families that differ in expressivity, inductive bias, and data requirements. Linear models, including linear regression and logistic regression, provide simple and interpretable decision functions and often serve as strong baselines when informative features are available. Kernel methods extend linear predictors by implicitly mapping inputs into higher-dimensional feature spaces, enabling nonlinear decision boundaries that classify the data into different categories. Tree-based methods (e.g., decision trees, random forest, and gradient boosting) are strong baselines for structured/tabular data, particularly when the input features are informative and require limited representation learning. Across many of these approaches, performance depends heavily on feature engineering and preprocessing, which determine how task-relevant information is presented to the model.

Training these models is commonly framed as risk minimization, where parameters are chosen to minimize a task-dependent loss over training data. Standard losses include

mean squared error (MSE) for regression and cross-entropy (CE) for classification, typically combined with regularization to improve generalization. Regularization can be explicit (e.g., weight decay, sparsity penalties) or implicit through training procedures such as early stopping. Gradient-based optimization is central in many settings, either directly (for differentiable models) or as an enabling component (e.g., for differentiable surrogates), and practical training requires careful attention to optimization dynamics, hyperparameters, and validation protocols.

2.2. DEEP LEARNING

Deep learning (DL) is a subset of ML that differs primarily in that it learns intermediate representations jointly with the predictive function using multi-layer neural networks. Rather than relying on handcrafted features, neural networks perform end-to-end learning in which feature extraction and the end task (e.g., classification) are optimized together, often yielding hierarchical representations that improve performance when sufficient data and compute are available. An intuitive illustration of the differences in ML and DL pipelines is shown in Figure 2.1. The model architecture used in DL encodes domain structure: convolutional networks exploit locality and translation equivariance in images and audio frequency representations; recurrent neural networks and Transformer architectures model sequential dependencies in text and speech; and graph neural networks propagate information along edges to learn node-, edge-, or graph-level predictors. In practice, deep learning systems are trained with stochastic gradient methods (e.g., SGD or Adam) and stabilized via normalization and regularization techniques, with data augmentation frequently used to improve robustness and generalization.

Neural networks define parametric function classes by composing affine transformations with pointwise nonlinearities. In the feed-forward (multi-layer perceptron) case, a network with parameters θ implements a mapping $f_\theta : \mathbb{R}^d \to \mathbb{R}^k$, where d denotes the input dimension and k the output dimension. As illustrated in Figure 2.2, the model consists of layers of neurons, each computing an affine function of its inputs, followed by a nonlinear activation function. Concretely, for neuron j, receiving inputs $x \in \mathbb{R}^m$, the output is $y_j = \sigma(w_j^T x + b_j)$, where $w_j \in \mathbb{R}^m$ and $b_j \in \mathbb{R}$ are learnable parameters and $\sigma(\cdot)$ is a nonlinear function. The nonlinearity is essential; without it, the neural network would collapse into a single affine transformation and could not represent nonlinear decision boundaries.

Stochastic gradient descent (SGD) is the standard optimization procedure for training neural networks by minimizing an empirical risk objective over a dataset. Given a loss function $\ell(\cdot)$ and training set $\{(x_i, y_i)\}_{i=1}^N$, training is formulated as

$$\min_w \frac{1}{N} \sum_{i=1}^{N} \ell(f_w(x_i), y_i). \tag{2.1}$$

Instead of computing the gradient over all N samples at each step, SGD estimates the gradient using a mini-batch B_t and updates the parameters as

$$w_{t+1} = w_t - \eta_t \nabla_w \frac{1}{|B_t|} \sum_{(x,y)\in B_t} \ell(f_w(x), y). \tag{2.2}$$

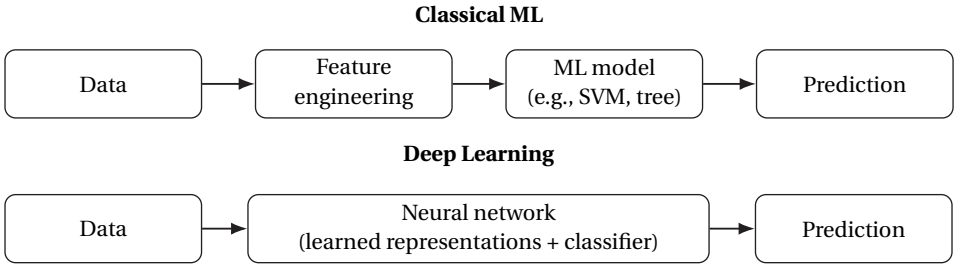


Figure 2.1: High-level contrast between classical machine learning pipelines (explicit feature engineering) and deep learning (end-to-end representation learning).

where η_t is the learning rate. Learning rate is a hyperparameter that controls how much the model's parameters are changed in each step. This stochastic approximation (update parameters based on mini-batches of data) substantially reduces computation and often improves generalization, but makes training sensitive to hyperparameters (learning rate schedule, batch size, momentum) and to regularization and normalization choices.

Multi-layer feedforward networks are a class of universal approximators, meaning that they can approximate quite well nearly any function of interest to a desired level of accuracy [10]. A more precise definition is: for many choices of activation function (e.g., Sigmoid, Tanh, ReLU; more generally, non-polynomial activations), a sufficiently wide feed-forward network with at least one hidden layer can approximate any continuous function on a compact subset of $\mathbb{R}$ to arbitrary accuracy (in the uniform norm). Variants extend approximation guarantees to broader function classes under L_p metrics. It is worth noting here that (i) the theorem is about the existence of parameters, not about whether standard training procedures will find them; (ii) "any function" is not literal, as regularity assumptions and the choice of norm/domain are part of the statement.

Universal approximation is important, but we should not ignore capacity and scalability. The number of parameters required to achieve a given approximation error can grow rapidly with input dimension and with the complexity of the target function, and different network architectures trade off width, depth, and inductive bias (sets of assumptions that restrict which function a model tends to learn). For ReLU networks in particular, the function class is piecewise linear; increasing depth/width increases the number of linear regions and thus expressive power, but it does not guarantee learnability, robustness, or generalization. In backdoor settings, these distinctions matter because "expressivity" can support both benign generalization and malicious conditional behaviors, while stealth and trigger robustness are shaped by the interaction between model capacity, optimization dynamics, and the data distribution.

2.2.1. CONVOLUTIONAL NEURAL NETWORKS

A very popular deep learning architecture (until at least the introduction of the transformer architecture) is the Convolutional Neural Networks (CNNs). Convolutional neural networks are a class of neural networks designed for data with spatial or temporal structure (e.g., images, spectrograms). Their core operation is the convolution layer, which applies a small learnable filter (kernel) across the input using local receptive fields and

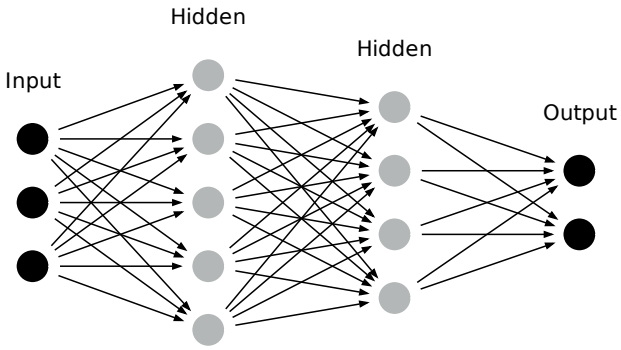


Figure 2.2: Example neural network with 3 inputs, 2 hidden layers, and 2 output classes

weight sharing. Compared to fully connected layers, convolution reduces parameter count and induces an inductive bias toward locality and translation structure: the same pattern detector (i.e., convolution filter/kernel) is reused at every location, producing a feature map that is approximately translation-equivariant. Practical CNN layers are defined by hyperparameters such as kernel size, stride, or padding, which control the resolution and receptive field of the resulting feature maps.

CNNs typically stack convolution layers with pointwise nonlinearities (e.g., ReLU) and sometimes pooling layers to downsample spatial resolution while increasing channel depth. Early layers often learn edge- and texture-like detectors, while deeper layers combine these into higher-level patterns. For classification, the final stages map learned feature maps to logits using a small "head" (e.g., global average pooling plus a linear layer, or a compact fully connected block). In modern practice, CNNs are trained end-to-end with gradient-based optimization, and architectural choices (normalization, residual connections, data augmentation, dropout) strongly influence training stability and generalization.

In Figure 2.3, we illustrate a convolution operation on a 5×5 input array using a 3×3 filter. To compute a single output element, the filter is positioned over a 3×3 input patch. The output is obtained by elementwise multiplication of the filter and the patch, followed by summation of the resulting products. The filter slides over all the possible positions of the input matrix.

In Figure 2.4, we demonstrate how a max-pooling layer downsamples the input feature map. In this example, we start from a 4×4 matrix, and we use a 2×2 kernel with a step (stride) of 2, which means that the kernel is moved by two positions every time. Thus, from the upper left 2×2 sub-matrix, the value 6 is chosen, from the upper right the value 2, from the lower left 3, and from the lower right 7. Pooling layers aim to make the model less dependent on the position of features inside the pooling window (translation invariance).

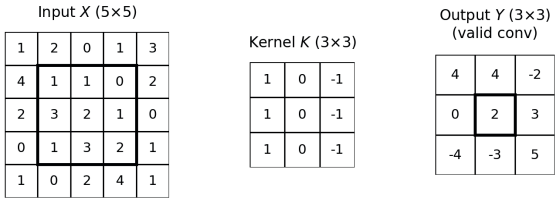


Figure 2.3: A numeric example of a 2D convolution operation. The mask hovers over the input array, producing a new one. To calculate the middle element of the output array, we do the following calculation: $(1 \times 1) + (3 \times 1) + (1 \times 1) + (0 \times 1) + (0 \times 2) + (0 \times 3) + (-1 \times 0) + (-1 \times 1) + (-1 \times 2) = 2$. The remaining elements are calculated in a similar way.

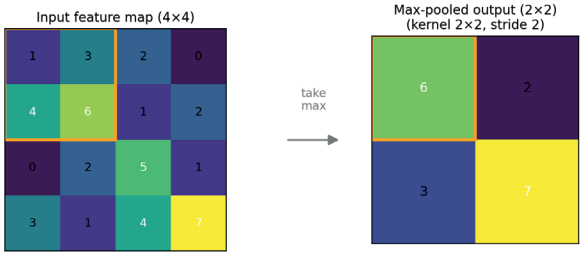


Figure 2.4: A numeric example of a max-pooling layer. A 2×2 window selects the maximum value of a 4×4 layer.

2.3. FEDERATED LEARNING

Federated learning (FL) [18] is a distributed training paradigm where multiple clients collaboratively learn a shared model without sharing their raw training data. As shown in Figure 2.5, a server maintains global parameters w and iteratively coordinates training rounds: it broadcasts w to a subset of clients, each client performs local optimization on its private dataset (e.g., several steps of SGD), and the resulting parameter updates are sent back to the server for aggregation, commonly via weighted averaging (FedAvg) proportional to each client's data size. FL aims to reduce data movement and address privacy or regulatory constraints, but it introduces challenges not present in centralized training, including heterogeneous (non-IID) data across clients, limited client participation, and new attack surfaces arising from potentially malicious or compromised clients.

FL can be applied in two cases: the cross-device scenario and the cross-silo scenario. In the first case, we assume that there are thousands of clients (e.g., mobile phones or edge IoT devices) where they all collaborate together [11]. One of the first commercial applications with a big client base is GBoard, the Google keyboard used in Android devices that leverages FL for more efficient word prediction [8]. In the cross-silo setting fewer parties are involved (up to 100 according to [8]). In this case the clients may be different organizations such as different hospitals training a common model without sharing sensitive data about their clients.

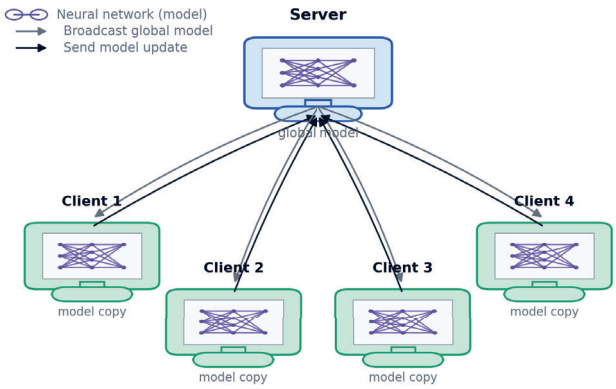


Figure 2.5: Federated learning schematic

2.4. GRAPH NEURAL NETWORKS

Graph data represents entities and their relationships as a graph $G = (V, E)$, where nodes V correspond to objects (e.g., users, devices, papers, accounts, molecules) and edges E encode interactions or relations (e.g., friendships, transactions, citations, chemical bonds). Common applications include social and communication networks (community detection, recommendation), fraud and anomaly detection (transaction graphs), knowledge graphs (link prediction and reasoning), molecular property prediction (drug discovery).

A graph $G = (V, E)$ is commonly represented by an adjacency matrix $A \in \{0,1\}^{|V|\times|V|}$, where $A_{wv} = 1$ indicates an edge between nodes u and v (or, more generally, A_{wv} can store an edge weight). A is symmetric if the graph is undirected and non-symmetric otherwise. In addition to the node connectivity encoded by A, each node v typically has a feature vector $x_v \in \mathbb{R}$. Stacking these features gives a node-feature matrix $X \in \mathbb{R}^{|V|\times d}$.

A Graph Neural Network (GNN) is a neural architecture designed to operate directly on graph-structured data by learning representations for nodes, edges, or entire graphs that incorporate both attribute information and graph connectivity. Most modern GNNs follow a message-passing formulation: at each layer, a node v receives "messages" computed from the current representations of its neighbors $u \in \mathcal{N}(v)$, aggregates them with a permutation-invariant operator (e.g., sum, mean, or max), and combines the result with its own representation to produce an updated embedding. Repeating this process for multiple layers propagates information across multi-hop neighborhoods, enabling predictions that depend on relational context rather than only local features. This framework covers a broad family of models, including graph convolutional networks (GCNs), and graph attention networks (GATs).

A GCN is a neural network that generalizes convolution/aggregation to non-Euclidean structure like graphs by repeatedly combining a node's representation with those of its neighbors ("message passing"). In its standard form, each layer updates node features via a normalized neighborhood aggregation followed by a learned linear map and nonlinear-

ity. Each layer's propagation rule is [13]:

$$H^{l+1} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}). \tag{2.3}$$

In this equation $\tilde{A} = A + I_N$, where A is the adjacency matrix, and I_N is the diagonal matrix that represents self-connections in each node. $\tilde{D} \in \mathbb{R}^{|V|\times|V|}$ is a diagonal matrix with $d_{ii} = \sum_j \tilde{A}_{ij}$ and $d_{ij} = 0$ where $i \neq j$. Thus, $\tilde{D}$ is the degree matrix of $\tilde{A}$. $W^{(l)}$ is the lth layer's trainable weights, $\sigma()$ is a non-linear activation function and $H^{(l)}$ is the activation map of the lth layer.

A Graph Attention Network (GAT) extends neighborhood aggregation by assigning different importance to neighboring nodes through a learnable attention mechanism [19]. Given node feature vectors $\mathbf{h}_i$ and $\mathbf{h}_j$, the importance of node j features to node i is initially computed as

$$e_{ij} = a(W\mathbf{h}_i, W\mathbf{h}_j), \tag{2.4}$$

where W is a trainable weight matrix and $a(\cdot,\cdot)$ is a learnable attention mechanism. The coefficients are then normalized across the neighborhood of node i using the softmax function:

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k\in\mathcal{N}_i}\exp(e_{ik})}. \tag{2.5}$$

In the original GAT formulation, the attention mechanism is implemented as a single-layer feed-forward network parameterized by a trainable vector $\mathbf{a}$, with a LeakyReLU (LR) activation [20]. Thus, the normalized attention coefficients are given by

$$a_{ij} = \frac{\exp\left(\mathrm{LR}\left(\mathbf{a}^T[W\mathbf{h}_i \| W\mathbf{h}j]\right)\right)}{\sum_{k\in\mathcal{N}i}\exp\left(\mathrm{LR}\left(\mathbf{a}^T[W\mathbf{h}_i \| W\mathbf{h}_k]\right)\right)}. \tag{2.6}$$

The updated representation of the feature vector of node i is obtained as a weighted aggregation of its neighboring node features:

$$\mathbf{h}'_i = \sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}W\mathbf{h}_j\right), \tag{2.7}$$

where $\mathcal{N}_i$ denotes the neighborhood of node i, α_{ij} represents the learned importance of node j to node i, and $\sigma()$ is a non-linear activation function. In general, GATs employ multiple attention heads, allowing the model to learn different patterns of interaction between neighboring nodes and improving the stability of the learned representations.

2.4.1. APPLICATIONS

Graph neural networks are commonly evaluated on *node classification* and *graph classification* tasks, which differ in what constitutes a training example and what is predicted. In node classification, a single graph $G = (V, E)$ is given along with node features X, and the goal is to predict a label y_v for each node $v \in V$. Training typically uses labels for a subset of nodes while message passing leverages the full connectivity to propagate contextual information across neighborhoods. An example application of node classification is

financial fraud detection on a transaction graph where nodes represent accounts and edges represent transactions [5].

In graph classification, each example is an entire graph G_i (often with node and/or edge features), and the goal is to predict a graph-level label y_i. (e.g., molecular property prediction); a GNN first computes node embeddings via message passing and then aggregates them into a single graph representation using a readout function (e.g., sum/mean pooling or attention) before applying a classifier. These tasks emphasize different aspects of GNN modeling: node classification exploits shared structure within a graph, while graph classification relies on permutation-invariant pooling to summarize graph-wide patterns. For example if we represent a program's structure or its execution flow as a graph then we can use a GNN to identify whether the program is malware [2].

2.5. BACKDOOR ATTACKS IN DEEP LEARNING

A backdoor inserts a secret functionality into a deep learning model that is activated through malicious inputs. The model's performance on clean inputs should not be affected by the backdoor insertion. Such functionality can be inserted into the model through data poisoning [7], code poisoning [1], or direct modification of the model weights [9]. In this thesis, we are working mostly with training-time data poisoning backdoor attacks.

A malicious input has been modified by the adversary and contains the backdoor trigger. The adversary is free to choose how the trigger looks and what its properties are. Attackers tend to use subtler, imperceptible triggers to make the attack harder to spot. In recent years, backdoors have been introduced in a large variety of applications. The first backdoors in deep learning were introduced in computer vision [7, 4], and we have since seen backdoor attacks in various domains. A few examples include domains like text [3], audio [14, 15], Graph Neural Networks [21, 23], 3D point cloud classifiers [22], semi-supervised learning [24], reinforcement learning [12], or malware detection [16].

There are various elements that characterize a backdoor attack [6, 17]. We have summarized these elements in the following list:

- ***Trigger***: The trigger is a property that the model's input has and activates the backdoor. Attackers can arbitrarily choose their triggers; therefore, the related literature [17] reports a wide variety of triggers.

- ***Target Class***: In classification tasks, the backdoor usually aims at a targeted misclassification. The target class is the class that the backdoored model outputs when the backdoor is activated. For example, a backdoored face-recognition-based access control system could grant an employee access to the CEO's office.

- ***Poisoning Rate***: In data poisoning scenarios, this variable shows the amount of data that the adversary altered ("poisoned") to insert the backdoor into the model.

- ***Dirty-label Attack***: In this scenario, the adversary inserts the trigger to training samples that originally belong to different classes and also changes their label to the target class.

- ***Clean-label Attack***: In this scenario, the adversary inserts the trigger into samples from the target class without changing their labels.

- ***Source-agnostic Backdoor***: A source-agnostic backdoor can be activated by poisoned inputs that originally belonged to any class.

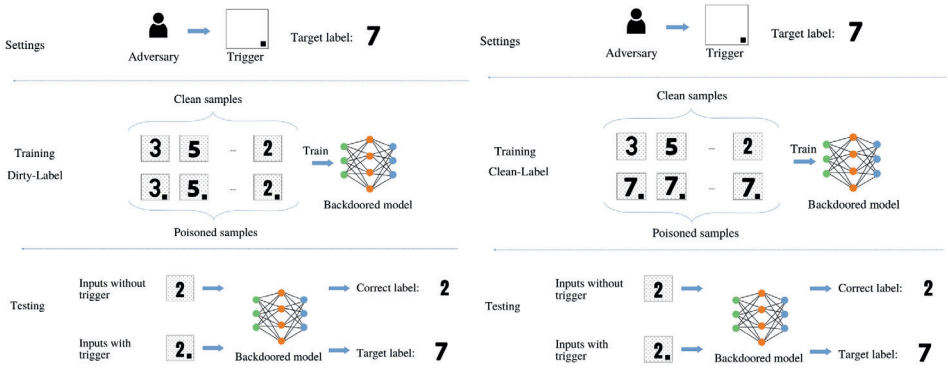
- ***Source-specific Backdoor***: A source-specific backdoor is activated only by poisoned inputs that originally belonged to specific classes. A sample that is not originally from the attacker-chosen classes cannot activate the backdoor even if the trigger is inserted into it.

- ***Static/Dynamic Trigger***: A static trigger is a trigger that does not change based on the sample that is inserted into. It has the same characteristics every time. In computer vision, a static trigger could be a patch whose pixels always have the same colors and are placed in the same positions. In audio, a pulse at a specific frequency can be considered static, since its representation in the frequency domain is just a component at that frequency. A dynamic trigger changes based on the input applied to it. In computer vision, a warping-based trigger is dynamic as it just adds a warping effect to the poisoned image.

- ***Digital/Physical Attack***: In the backdoor literature and in adversarial machine learning in general, a digital attack is tested only digitally by just providing the model with the adversarial input. In a physical attack, the attacker tests the system in a more realistic setup, manipulating the model's input in the physical world (e.g., by changing what the cameras capture in a computer vision application).

- ***Input/Feature/Parameter space***: In the backdoor literature, we often examine the effect that the trigger has on the model's inputs and the model's latent space representations. Additionally, depending on the used threat model, we can examine the backdoor's effect on the model's parameters.

- ***Stealthiness***: Stealthiness in backdoor attacks refers to the extent to which the presence of a trigger in a poisoned sample remains difficult to perceive or detect. Its characterization may differ across domains such as vision, audio, and text. At the input level, stealthiness is related to the perceptibility of the trigger and to the magnitude of the perturbation required to insert it. At the feature level, it can also refer to how much the trigger alters the model's internal representations. These forms of stealthiness can affect the ability of a backdoor attack to evade backdoor defenses. For example, visually imperceptible triggers may be less likely to be identified through manual data inspection or input-level anomaly detection. Stealthiness can therefore be evaluated using both instance-level measures, such as similarity between clean and poisoned samples, and distribution-level measures, such as the ability of a detector to distinguish poisoned samples from clean ones.



(a) *Dirty-label* backdoor attack. (b) *Clean-label* backdoor attack.

Figure 2.6: Schematic overview of backdoor attack settings (target class is 7, trigger is a black square).

In Figure 2.6, we show an intuitive illustration of the standard data poisoning backdoor setup, covering both dirty-label (Figure 2.6a) and clean-label (Figure 2.6b) setups. In both cases, the example shows an MNIST-type classification of digits for its simplicity. Here, the trigger is a black square in the lower right corner, and the target class is 7. In the dirty-label setup, the adversary poisons samples from all the classes. On the contrary, for the clean label setup, the attacker poisons samples only from class 7.

The most commonly used metrics for measuring the performance of a backdoor attack are Attack Success Rate (ASR) and Clean Accuracy Drop (CAD). To calculate the ASR, we insert the trigger into the test set samples and then count how many times the backdoor is activated. Then we divide this number by the total number of poisoned samples. This number should be as close to 100% as possible. The CAD is the difference of the accuracy of a clean model on the clean test set from the accuracy of a backdoored model on the same clean test set. The CAD shows the performance drop of a backdoor model on clean inputs. As the backdoor should not affect the model's performance, this drop should be as small as possible.

REFERENCES

[1] Eugene Bagdasaryan and Vitaly Shmatikov. "Blind Backdoors in Deep Learning Models". In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 1505–1521. ISBN: 978-1-939133-24-3. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/bagdasaryan.

[2] Julian Busch et al. "NF-GNN: Network flow graph neural networks for malware detection and classification". In: *Proceedings of the 33rd international conference on scientific and statistical database management*. 2021, pp. 121–132.

[3] Xiaoyi Chen et al. "Badnl: Backdoor attacks against nlp models". In: *ICML 2021 Workshop on Adversarial Machine Learning*. 2021.

[4] Xinyun Chen et al. "Targeted backdoor attacks on deep learning systems using data poisoning". In: *arXiv preprint arXiv:1712.05526* (2017).

[5] Yingtong Dou et al. “Enhancing graph neural network-based fraud detectors against camouflaged fraudsters”. In: *Proceedings of the 29th ACM international conference on information & knowledge management*. 2020, pp. 315–324.

[6] Yansong Gao et al. “Backdoor attacks and countermeasures on deep learning: A comprehensive review”. In: *arXiv preprint arXiv:2007.10760* (2020).

[7] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. “Badnets: Identifying vulnerabilities in the machine learning model supply chain”. In: *arXiv preprint arXiv:1708.06733* (2017).

[8] Andrew Hard, Kanishka Rao, Rajiv Mathews, et al. “Federated learning for mobile keyboard prediction”. In: *arXiv preprint arXiv:1811.03604* (2018).

[9] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. “Handcrafted backdoors in deep neural networks”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 8068–8080.

[10] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural networks* 2.5 (1989), pp. 359–366.

[11] Peter Kairouz et al. “Advances and open problems in federated learning”. In: *Foundations and Trends® in Machine Learning* 14.1–2 (2021), pp. 1–210.

[12] Panagiota Kiourti et al. “Trojdrl: evaluation of backdoor attacks on deep reinforcement learning”. In: *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE. 2020, pp. 1–6.

[13] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: *ICLR*. 2017.

[14] Stefanos Koffas et al. “Can You Hear It? Backdoor Attacks via Ultrasonic Triggers”. In: *Proceedings of the 2022 ACM Workshop on Wireless Security and Machine Learning*. WiseML ’22. San Antonio, TX, USA: Association for Computing Machinery, 2022, pp. 57–62. ISBN: 9781450392778. DOI: 10.1145/3522783.3529523. URL: https://doi.org/10.1145/3522783.3529523.

[15] Stefanos Koffas et al. “Going In Style: Audio Backdoors Through Stylistic Transformations”. In: *ICASSP*. 2023.

[16] Chaoran Li et al. “Backdoor attack on machine learning based android malware detectors”. In: *IEEE Transactions on dependable and secure computing* 19.5 (2021), pp. 3357–3370.

[17] Yiming Li et al. “Backdoor learning: A survey”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[18] Brendan McMahan et al. “Communication-efficient learning of deep networks from decentralized data”. In: *Artificial intelligence and statistics*. PMLR. 2017, pp. 1273–1282.

[19] Petar Veličković et al. “Graph Attention Networks”. In: *ICLR* (2018). URL: https://openreview.net/forum?id=rJXMpikCZ.

[20] Petar Veličković et al. “Graph Attention Networks”. In: *International Conference on Learning Representations* (2018). accepted as poster. URL: https://openreview.net/forum?id=rJXMpikCZ.

[21] Zhaohan Xi et al. “Graph backdoor”. In: *30th USENIX Security Symposium (USENIX Security 21)*. 2021, pp. 1523–1540.

[22] Zhen Xiang et al. “A backdoor attack against 3d point cloud classifiers”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 7597–7607.

[23] Jing Xu, Minhui Xue, and Stjepan Picek. “Explainability-based backdoor attacks against graph neural networks”. In: *Proceedings of the 3rd ACM Workshop on Wireless Security and Machine Learning*. 2021, pp. 31–36.

[24] Zhicong Yan et al. “Deep neural backdoor in semi-supervised learning: Threats and countermeasures”. In: *IEEE Transactions on Information Forensics and Security* 16 (2021), pp. 4827–4842.

II

BACKDOORS IN SPEECH

3

Can You Hear It? Backdoor Attacks via Ultrasonic Triggers

This work explores backdoor attacks for automatic speech recognition systems where we inject inaudible triggers. By doing so, we make the backdoor attack challenging to detect for legitimate users and, consequently, potentially more dangerous. We conduct experiments on two versions of a speech dataset and three neural networks and explore the performance of our attack concerning the duration, position, and type of the trigger.

Our results indicate that less than 1% of poisoned data is sufficient to deploy a backdoor attack and reach a 100% attack success rate. We observed that short, non-continuous triggers result in highly successful attacks. Still, since our trigger is inaudible, it can be as long as possible without raising any suspicions making the attack more effective. Finally, we conduct our attack on actual hardware and saw that an adversary could manipulate inference in an Android application by playing the inaudible trigger over the air.

This chapter has been published as S. Koffas, J. Xu, M. Conti, S. Picek "Can you hear it? backdoor attacks via ultrasonic triggers". *In Proceedings of the 2022 ACM workshop on wireless security and machine learning 2022 May 19 (pp. 57-62).*

3.1. INTRODUCTION

Automatic speech recognition has gained much attention in recent years as it can be a very efficient form of communication between people and machines. Voice assistants have already shown the accessibility and high recognition accuracy of automatic speech recognition. As researchers devote much of their effort to improving its performance, there are also works trying to understand how vulnerable is automatic speech recognition to adversarial attacks like backdoor attacks.

Backdoor attacks are a serious threat to neural networks. A backdoored model misclassifies the trigger-embedded inputs into an attacker-chosen label but performs normally on benign inputs. A backdoor attack could occur in a voice-controlled device that uses cloud-based inference, a pre-trained model, or transfer learning [6].

To protect systems against such threats, it is important to understand them and investigate in what scenarios they pose the biggest risk. An intuitive answer is in scenarios where the trigger cannot (or is difficult to) be noticed by legitimate users.

Liu et al. implemented a backdoor attack in automatic speech recognition by injecting background noise into the original audio sample and retraining the model to recognize the stamped audio as a specific word [9]. Zhai et al. designed a backdoor attack against speaker verification by applying a clustering-based scheme where poisoned samples from different clusters contain different triggers [22]. Xu et al. backdoored automatic speech recognition by generating a random sequence of data points for the trigger [21]. All these works inject unintelligible but audible triggers into the training dataset. As a result, humans can understand the differences between the trigger-embedded and the clean inputs.

In this work, we consider backdoors on automatic speech recognition using an inaudible trigger. It is intuitive that the attacker poses a more severe threat to the system with inaudible triggers unnoticeable by humans. To the best of our knowledge, there is no study exploring backdoor attacks with inaudible triggers in speech recognition systems. In this paper, we seek to bridge this gap. We did not exploit existing nonlinearities in the sound processing pipeline [23] as our goal is to shed light on new risks that can be introduced when a high sampling rate is used. While such sampling rates are not recommended for automatic speech recognition [5], they are still allowed by public speech-to-text APIs [5, 10]. Our main contributions are:

- We propose a backdoor attack on a speech recognition system using an ultrasonic pulse as the trigger.
- We systematically evaluate the impact of various trigger characteristics on the performance of backdoor attacks on automatic speech recognition.
- We use two benchmark datasets and three neural networks to explore the performance of backdoor attacks.
- We show that our attack is effective with real hardware by attacking an Android application.

To foster reproducibility, our code is publicly available[1].

[1] https://github.com/skoffas/ultrasonic_backdoor

3.2. PRELIMINARIES

3.2.1. AUTOMATIC SPEECH RECOGNITION

Computers use automatic speech recognition to understand human speech and translate it into text. Modern automatic speech recognition uses deep learning. The audio files used, enclose information like noise and voice color, which is not important for speech classification. Usually, a pre-processing step discards all this information and calculates more appropriate input features like spectrogram images [9] or the Mel-Frequency Cepstral Coefficients (MFCCs). We use the MFCCs, which are 1) rather accurate as they emulate the functionality of the human vocal system [8] and 2) widely used [21, 15].

3.2.2. BACKDOOR ATTACKS

In a backdoor attack, the adversary aims to train a neural network that correctly solves the desired task on expected data but exhibits malicious behavior once presented with a certain trigger [6]. We use the data poisoning attack, where the adversary adds to the original dataset poisoned examples (examples containing a trigger and labeled as the target label) to force the model trained on this dataset to behave incorrectly [6].

Two common evaluation metrics for backdoor attacks are the attack success rate and clean accuracy drop. The attack success rate (ASR) is the fraction of successfully triggered backdoors over a set of poisoned inputs. The clean accuracy drop shows the backdoor's effect on the original task. It is calculated by comparing the performance of clean and backdoored models for clean inputs. We compute both metrics on the entire test set.

3.3. INAUDIBLE BACKDOOR ATTACK

3.3.1. TRIGGER

To the best of our knowledge, we are the first to experiment with a stealthy trigger in the inaudible range (> 20kHz). Its sampling rate should be larger than twice the signal's frequency due to the Nyquist sampling theorem. We select the trigger's sampling rate to be 44.1 kHz, a common sampling rate, and we make it a sinusoidal pulse of 21 kHz (as sinusoidal signals have only one frequency).

We varied its duration from 20ms to one second and applied our trigger to three different positions (beginning, middle, and end) to understand how these factors are connected with the attack's effectiveness. Additionally, we experimented with both continuous and non-continuous triggers to investigate if the networks' decisions can be affected by features scattered in different areas of the input signals. To create a non-continuous trigger of x ms, we distribute in every 1-second audio sample five triggers of $\frac{x}{5}$ms. Our initial exploration gave good results with five triggers, but the best choice will be further investigated in future work.

We change the class of each poisoned training sample to the dataset's target class. We chose different target classes for each dataset version (see Section 3.4.1) to verify the attack's effectiveness in different settings. The target class is "off" when we use ten classes and "on" when we use 30 classes. We poison the first N samples of the, dataset [4], which is technically similar to random poisoning because the dataset is already shuffled before the poisoning.

3.3.2. THREAT MODEL

We follow a gray-box data poisoning attack. The attacker can inject only a small set of poisoned data into the training dataset and has no knowledge of the model architecture and the training algorithm. This threat model is realistic as modern datasets are usually based on crowdsourcing both for their creation and validation [2, 20]. Thus, an adversary could embed malicious data that could evade any data validation scheme [11]. Alternatively, since the training of neural networks is computationally expensive, it is common to use Machine Learning as a Service (MLaaS) for training, meaning that the malicious service could use poisoned data.

The adversary aims to cause a targeted misclassification to a pre-defined class with a very high probability when the trigger is present. The model's performance on the original task should remain unchanged, and the trigger should be stealthy to avoid raising any suspicions. Finally, for settings where the signal sampling rate is lower than 42 kHz, the attack would only be possible under a different threat model, because our trigger requires a sampling rate of at least 42 kHz to be reliably recognized. In this case, before poisoning, the adversary would need access to and modify the sampling rate of the entire training dataset, even though the trigger is added to only a small fraction of it. This requirement results in a white-box threat model with respect to data access and model architecture.

3.4. EXPERIMENTAL SETUP

3.4.1. DATASET AND INPUT FEATURES

We use two different versions of the Speech Commands dataset. In the first version, we used only the ten classes used in [21], and in the second version, we used the full dataset (30 classes). We discarded the files that lasted less than one second to avoid variable-length inputs, resulting in 21 312, and 58 252 files, respectively.

As we discussed in Section 3.3.1, the trigger's sampling rate is 44.1 kHz. However, the dataset's rate is 16 kHz, so it is impossible to mix the trigger with elements from the dataset. For that reason, we up-sampled the dataset's audio files from 16 kHz to 44.1 kHz. This is not a typical sampling rate for speech recognition systems [5], but still supported by popular public APIs [5, 10].

We use the MFCCs as input features. In particular, we used 40 mel-bands, a step of 10ms, and a window length of 25ms that is very common in related works [19, 22].

3.4.2. NEURAL NETWORK ARCHITECTURES

We use two CNNs and one LSTM. The first CNN was also used in [15]. However, we added two dropout layers, one in the penultimate layer (40%) and one right before the flatten layer (50%), which gave a performance boost of around 3%. The second CNN is deeper and was used in [9], and the LSTM was introduced in [1].

All the models perform similarly as the loss function is around 0.3 when the training ends. However, their capacity is different as the first CNN has 284 778 trainable parameters, the second CNN has 5 975 882, and the LSTM has 180 569. In this way, we can investigate whether both simple and more complex networks are susceptible to inaudible backdoor attacks. We also investigate whether the backdoor attack behaves differently for different neural networks (CNNs and LSTMs). We use the Adam optimizer with a learning rate of

0.0001 to train our models and L_2 regularization in every convolution filter in both CNNs. 80% of the initial dataset was used for training/validation, and the remaining 20% for testing. The training dataset is again split 80%/20% for training and validation.

After carefully examining the training and validation loss functions, we saw that the CNNs could always converge before 300 epochs and the LSTM before 100. We used Tensorflow's early stopping callback with patience 20, which stops the training if the validation loss is not improved for 20 epochs returning the weights of the best-performing model.

The arithmetic mean of the epochs among all the experiments we run is 220.8 (± 24.97) for the small CNN, 182.4 (± 31.39) for the large CNN, and 70.9 (± 12.70) for the LSTM for the ten dataset classes and 250.5 (± 32.96), 102.5 (± 29.46), and 83.3 (± 13.16) for the 30 classes. We run each experiment 15 times to eliminate any randomness introduced by the algorithmic randomness (e.g., random initialization and stochastic gradient descent). All the models were trained with Tensorflow 2.5 on NVIDIA RTX 2080 Ti.

3.5. EXPERIMENTAL RESULTS

This section presents a systematic evaluation of our inaudible backdoor attack. We investigate how the ASR is affected by various backdoor characteristics like the number of poisoned samples, the trigger's duration, and the trigger's position in time.

The triggers that we tried can be split into two groups: the long (250ms, 500ms, 750ms, and 1 000ms) and the short triggers (20ms, 40ms, 60ms, and 80ms). In this way, we aim to see if the attack is successful, even if only a few MFCC frames are affected by our trigger. Additionally, we applied our trigger in different positions to see if our models give more attention to the windows that usually contain speech.

First, we need to define the poison percentage that leads to an effective attack. Figure 3.1 shows ASR as a function of the poison percentage for each architecture. In these experiments, we used a trigger of 20ms in the middle of each signal, which is short and makes the attack very challenging. When we poison 200 samples or more for both CNNs, ASR is close to optimal even with this short trigger. These results are further improved by longer or non-continuous triggers and different trigger positions. Thus, we used a smaller poison percentage to keep any differences between different setups distinguishable. In particular, we poisoned at most 80 (0.59% for the ten classes and 0.21% for the 30 classes) training samples in the experiments with the CNNs. The ASR is lower for the LSTM architecture. The attention layer can give useful insights into a network's choices [1]. In Figure 3.2, we show the logarithm of the attention weights for two backdoored models (20ms trigger in the middle used for the backdoor) when a poisoned input is used. We see that the attention layer does not recognize the trigger when 200 poisoned samples are used. This indicates that the attention layer is more robust to the backdoor attack suggesting that unless a smart position/duration/type of a trigger is used, the backdoor attack on LSTM could be rather difficult.

Roadmap. In our results, we show ASR vs. poisoning rate. Each bar chart consists of three subplots, one for every trigger position. Each bar in these plots represents a trigger of a different duration. Furthermore, to better view the attack effectiveness for different trigger positions, we plot the average ASR for a different number of poisoned samples in each trigger position. Each of these graphs consists of four lines, one for each position

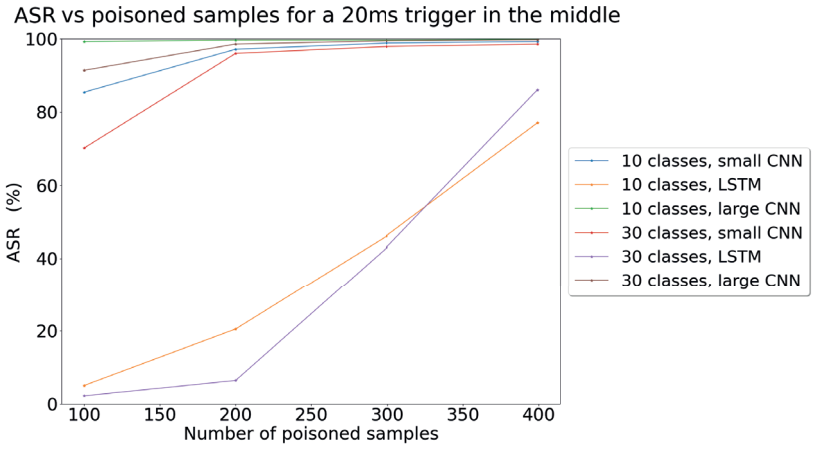


Figure 3.1: ASR vs. trigger samples for more than 80 poisoned samples and CNNs.

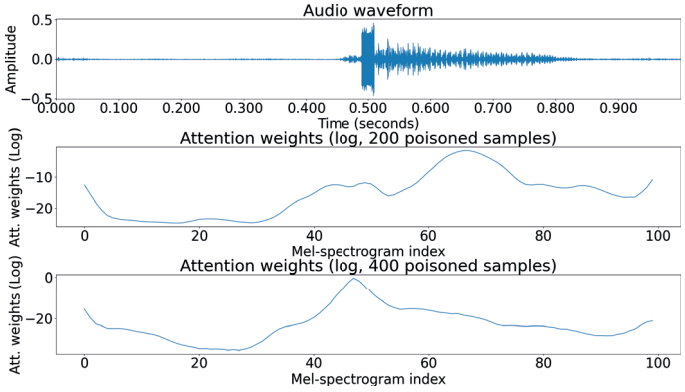


Figure 3.2: Attention weights for poisoned input ("on").

(three lines) and one for the non-continuous trigger. We plot only the results for the 30 classes as there are no significant differences between the two versions of the dataset for each architecture.

3.5.1. EFFECT ON CLEAN ACCURACY

A backdoor attack should remain hidden to avoid raising any suspicions. Thus, we need to verify that our attack does not affect the model's performance on the original task. In Tables 3.1 and 3.2, we compare the clean model's accuracy on the original task with the accuracy that backdoored models have. To obtain the values for the poisoned models, we group them based on the number of poisoned samples used and take the average (arithmetic mean) of their clean accuracy for all the different triggers used.

The differences for the original task are negligible between the clean and the backdoored CNNs (Table 3.1). The backdoored models perform slightly better in the original

task when we use ten classes. This behavior indicates that backdoored training data could improve the model's generalization, probably serving as a regularization factor. However, the differences are small, and further investigation is required to confirm our observations. When we use the full dataset, the backdoor insertion drops the clean accuracy in all cases for both CNNs (somewhat larger accuracy drop for large CNN). Again, this is aligned with our previous observations, as more classes (and a larger dataset) also represent a more difficult task for neural network architectures to learn.

Table 3.1: Clean accuracy comparison for the original task for clean and poisoned models (CNNs).

			Number of poisoned samples			
model	classes	original	20	40	60	80
small CNN	10	90.13 (± 0.533)	90.22 (± 0.127)	90.19 (± 0.086)	90.16 (± 0.108)	90.14 (± 0.091)
small CNN	30	87.58 (± 0.446)	87.52 (± 0.114)	87.46 (± 0.161)	87.49 (± 0.083)	87.47 (± 0.118)
large CNN	10	95.71 (± 0.424)	95.82 (± 0.122)	95.82 (± 0.139)	95.85 (± 0.133)	95.85 (± 0.112)
large CNN	30	95.14 (± 0.539)	94.82 (± 1.527)	94.61 (± 2.609)	94.24 (± 2.292)	93.52 (± 3.621)

The LSTM behaves oppositely. In Table 3.2, the clean accuracy is dropped in all cases when the ten classes are used and increased when the entire dataset is used. This indicates that the LSTM built different models in each case, and it utilized the model capacity better for ten classes setting.

Table 3.2: Clean accuracy comparison for the original task for clean and poisoned models (LSTM).

			Number of poisoned samples			
model	classes	original	40	80	120	160
LSTM	10	91.27 (± 1.217)	91.09 (± 0.354)	91.13 (± 0.29)	90.97 (± 0.459)	90.72 (± 0.359)
LSTM	30	91.45 (± 0.748)	91.76 (± 0.198)	91.63 (± 0.208)	91.59 (± 0.24)	91.77 (± 0.241)

3.5.2. ON THE EFFECT OF TRIGGER PROPERTIES ON ASR

CNNs

In Figure 3.3, we show the results for different triggers for the small CNN and the full dataset. The attack is more successful for the large CNN as it is a deeper architecture and is more capable of learning features from only a few training samples. However, we observe some similarities in the attack's behavior in both architectures, and for that reason, we omit the corresponding graph for the large CNN. In particular, ASR increases linearly as the trigger's duration increases, especially in cases when it is not already high. We also see that long triggers result in a high ASR even if only 20 poisoned samples are used. Additionally, ASR for short triggers is low when using only 20 poisoned samples but is significantly larger when using 80 samples. This shows a strong connection between ASR and the number of poisoned samples used for the attack.

In Figure 3.4, we show the mean ASR for different backdoored models for the small CNN when the full dataset is used. Again, we omit the corresponding graph for the large CNN. We saw in both CNNs that the position is not important when a long trigger is used. However, the attack effectiveness varies for small triggers. For the small CNN, the non-continuous trigger is substantially more effective than the continuous ones, and the

middle of the signal is the worst position for the trigger. The attack is more effective when more MFCC windows are affected and the trigger does not overlap with actual speech. In the large CNN, the non-continuous trigger outperforms the continuous ones only when the trigger lasts for 20ms. The network's complexity allows effective backdoor insertion even with short triggers longer than 20ms.

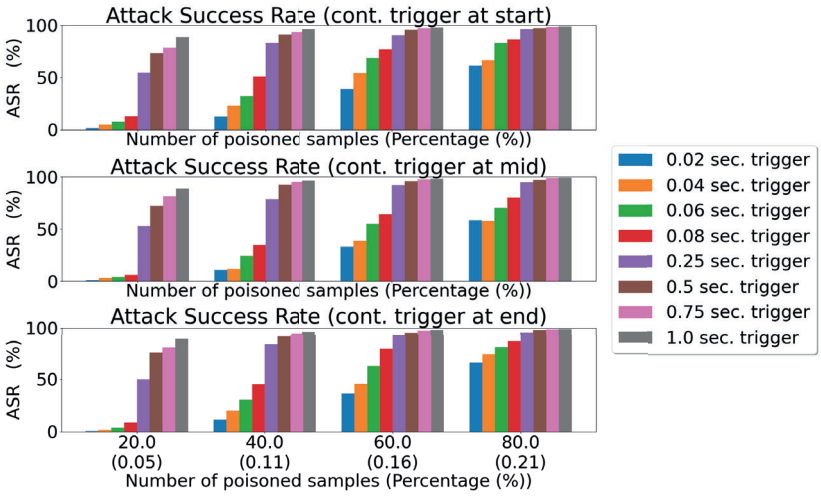


Figure 3.3: ASR vs. poisoned samples for all the trigger positions and sizes for 30 dataset classes (small CNN).

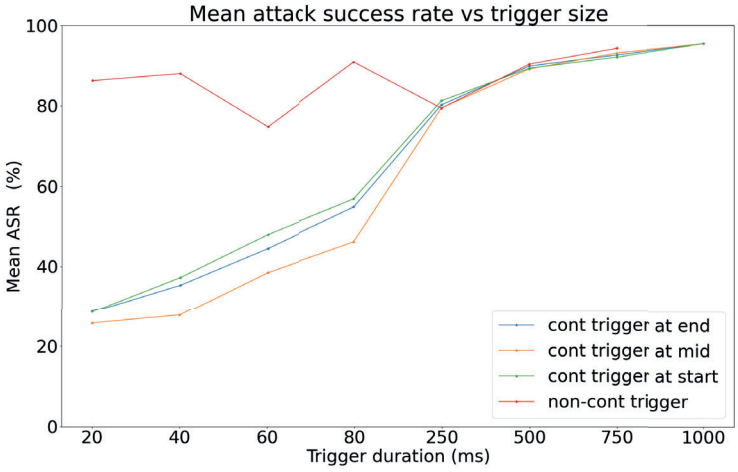


Figure 3.4: Mean ASR vs. trigger size for 30 dataset classes (small CNN).

LSTM

In Figure 3.5, we show the results for each different trigger for the LSTM network for 30 classes. The ASR is lower for this architecture even though we use more poisoned samples. As shown above, the attention layer cannot be affected easily by small variations inside the audio file. However, similarly to CNNs, there is a linear correlation between the trigger's duration and ASR. Again, for long triggers, we see less difference in ASR, especially when using more poisoned samples.

In Figure 3.6, we show the mean ASR among different backdoored models for the LSTM network when the full dataset is used. We see that the trigger at the end of the signal is the most effective continuous trigger in both dataset versions. Additionally, non-continuous triggers clearly outperform continuous ones.

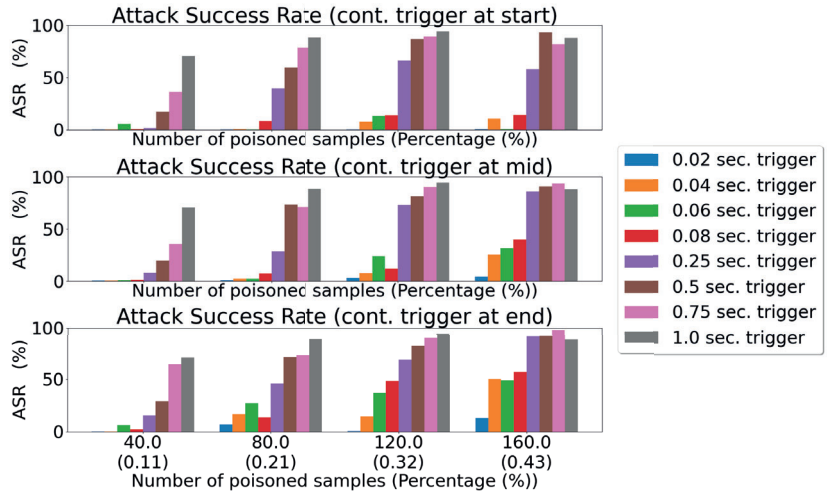


Figure 3.5: ASR vs. poisoned samples for all the trigger positions and sizes for 30 dataset classes (LSTM).

3.5.3. ATTACK ON ANDROID APPLICATION

Many microphones are inherently capable of recording audio signals in the near ultrasonic range (e.g., 18 to 22 kHz) [14]. To this end, we played our trigger through a VLC media player on a Linux laptop and verified through Spectroid [13] that a signal of 21kHz was played even if we could not hear anything. This experiment showed that the laptop speaker could transmit sounds in the inaudible human range, and the smartphone microphone could interpret them. Additionally, we verified that the inaudible trigger was successfully generated, which is the first step for a practical attack.

As a second step, we applied our attack on an Android speech recognition application. Our application is based on a TensorFlow Lite official example [16] that recognizes ten spoken words from the Speech Commands dataset. However, we modified its preprocessing pipeline to calculate the audio's MFCCs and used a poisoned model for the inference. We only run this experiment with the small CNN, but we believe this attack

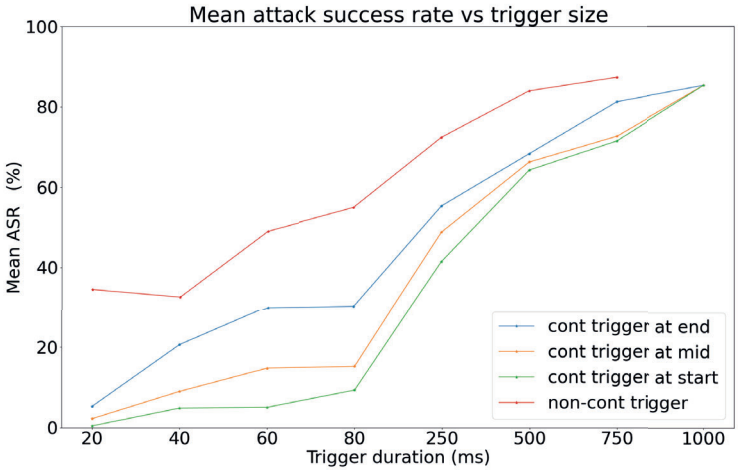


Figure 3.6: Mean ASR vs. trigger size for 30 dataset classes (LSTM).

will be effective with all three architectures. We decided to use the small CNN because it uses less memory, while the ASR with the LSTM showed worse performance, making it less attractive architecture. We used our strongest setup, i.e., 80 poisoned samples and a 1-second trigger to model our adversary. The application records 1-second clips and shows the keywords that are recognized with 50% probability or more.

We installed our application on a smartphone that ran Android 7.0 (Huawei P9 Lite). We played our inaudible trigger both from a Linux laptop (Lenovo t460s) and low-end home cinema speakers (Philips HTS7200). We run two experiments for each sound device. In the first, we played our trigger in silence to ensure that the backdoor was successfully activated when no sounds were present. In the second, we said words of the nine possible classes (omitted the target class) in random order while the trigger was played by either the laptop or the speakers. In each experiment, we placed the smartphone in four different locations to see how the attack accuracy is related to the distance from the sound source. The first location is adjacent to the sound source (0m), and the remaining ones are 0.5m, 1m, and 1.5m away from it. We run each experiment 15 times to have a more robust view of the attack's behavior.

Using Spectroid [13], we saw that for both devices, the trigger's intensity is around −43dB when the phone is adjacent to the sound source. However, as the distance between the speaker and the phone grows, this number drops significantly faster for the laptop experiments. Additionally, the trigger's intensity is lower than the normal voice for all the positions we tried. As a result, the backdoor was successfully triggered only in silence as its volume was not enough to overcome our speech. In particular, ASR is high only when the distance is 0m if the trigger is played through the laptop. On the other hand, if the trigger is played from the home cinema speakers, ASR is high (> 80%) for distances up to 1m. At 1.5m, the backdoor is still activated but with a lower ASR. We saw that ASR is further increased when the user does not speak loud enough or the phone is not close

enough to the speaker's mouth. We concluded from these experiments that this attack poses a real threat as an adversary with stronger equipment could be more effective.

3.5.4. GENERAL OBSERVATIONS

Based on the conducted experiments, we list several findings that were consistently observed.

- The attack success rate is linearly correlated with the trigger's duration. Our models can learn longer triggers easier as more windows in the MFCC calculation are affected.
- Our attack is inaudible, so an adversary could use a 1-second trigger for a powerful attack with a very low poisoning rate.
- If the adversary does not want to use a 1-second trigger (or any long trigger), non-continuous triggers represent the best option as they perform extremely well with short triggers.
- Different models behave differently for our attack. For example, the best ASR for the short continuous triggers and LSTM is at the end of the signal, but for the large CNN, the trigger at the beginning results in a slightly increased ASR. This indicates that the data poisoning backdoor attack could be used as a tool for AI explainability, as it can highlight how and what neural networks learn.
- The number of poisoned samples is clearly connected with ASR. Thus, an adversary should use as many samples as possible without risking a large performance drop for the original task to avoid raising suspicions.
- The backdoor in the Android application was successfully activated by playing the trigger from the laptop and low-end home cinema speakers within a 1.5m.

LIMITATIONS

While our attack showed very good performance, we believe there are also some limitations concerning inaudible backdoor attacks:

- We use MFCCS as a convenient representation of audio signals for machine learning, and we did not validate our results with other types of signal representations. Still, we do not consider this a significant limitation due to the widespread use of MFCCS and no intuitive reason that the attack would not work with different representations.
- As we use inaudible triggers, the sampling rate of the recorded signals should be larger than 40 kHz. If an automatic speech recognition system uses a lower sampling rate, our attack will not be effective. A sampling rate of 16kHz is preferred in automatic speech recognition [5], however popular public speech-to-text APIs allow higher rates [5, 10]. Additionally, different applications like music genre classification require high sampling rates [12], making our attack practical.
- Finally, our work did not consider any defense against the inaudible backdoor attack. While we show it is possible to achieve (almost) 100% ASR for various trigger positions and durations, even if using a small number of poisoned examples, different defenses could still work. Our trigger works for every source class in the dataset, and thus, an online defense like STRIP [4] could defend against it by superimposing different signals and checking the entropy of the model's output. Additionally, a

low-pass filter could make our attack ineffective as it could filter out frequencies above the human audible range. Such a filter could be either software-based or hardware-based. A software-based filter could introduce additional computational overhead, which may be unacceptable in a device with strict timing constraints. Moreover, a hardware-based filter may not be possible in any device as frequencies close to the ultrasonic range may be useful in some cases like music genre classification [12].

- When playing a poisoned audio sample, if the volume is too high, it is possible to hear a slight clicking sound in its beginning. This clicking sound is connected with the backdoor trigger. Its volume is very low compared to the main signal, so unless expecting it, it is hard to hear it. This sound is heard only in the trigger's beginning, and thus, we still consider it to be practically inaudible.

3.6. CONCLUSIONS AND FUTURE WORK

This paper explores how inaudible backdoor attacks on neural networks threaten automatic speech recognition. We use a dataset with ten or 30 classes and three neural networks (two CNNs and one LSTM). Furthermore, we investigate the influence of the position, duration, and trigger type. Our results suggest it is relatively easy to run an inaudible backdoor attack on automatic speech recognition where the attacker needs to poison only around 0.5% of the training dataset. Since the trigger is inaudible, it is possible to make it as long as the signal, making the attack even more powerful. We show that non-continuous triggers can significantly improve attack performance even in scenarios where short triggers are used. Consequently, short non-continuous triggers inserted in less than 0.5% of the training dataset can result in an attack success rate of more than 99%. Finally, we show that our attack is effective in the real-word by attacking an Android application.

As we did not consider countermeasures, this would be a natural extension of our work. Today, most of the defenses against backdoor attacks try to detect either the poisoned inputs or compromised models [7, 3]. As we limit our data poisoning to a small number of training examples, we believe such techniques would have difficulties detecting poisoning examples. Furthermore, some more recent defenses suggested retraining [17], which we consider to be a good choice against our attack. Still, since such works commonly assume an unlimited amount of clean data, they are not practical. We are aware of only one approach that does not have that limitation [18], but it can work for image processing only, making this defense not applicable against our attack.

REFERENCES

[1] Douglas Coimbra de Andrade et al. *A neural attention model for speech command recognition*. 2018. arXiv: 1808.08929 [eess.AS].

[2] Rosana Ardila et al. "Common Voice: A Massively-Multilingual Speech Corpus". In: *CoRR* abs/1912.06670 (2019). arXiv: 1912.06670. URL: http://arxiv.org/abs/1912.06670.

[3] Bryant Chen et al. “Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering”. In: *CoRR* abs/1811.03728 (2018). arXiv: 1811.03728. URL: http://arxiv.org/abs/1811.03728.

[4] Yansong Gao et al. “STRIP: A Defence against Trojan Attacks on Deep Neural Networks”. In: *Proceedings of the 35th Annual Computer Security Applications Conference*. ACSAC ’19. San Juan, Puerto Rico, USA: Association for Computing Machinery, 2019, pp. 113–125. ISBN: 9781450376280. DOI: 10.1145/3359789.3359790. URL: https://doi.org/10.1145/3359789.3359790.

[5] Google. *Speech-to-Text basics*. https://cloud.google.com/speech-to-text/docs/basics. [Online, Accessed: 8 March 2022]. 2022.

[6] T. Gu et al. “BadNets: Evaluating Backdooring Attacks on Deep Neural Networks”. In: *IEEE Access* 7 (2019), pp. 47230–47244. DOI: 10.1109/ACCESS.2019.2909068.

[7] Matthew Jagielski et al. “Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning”. In: *CoRR* (2018). arXiv: 1804.00308. URL: http://arxiv.org/abs/1804.00308.

[8] Uday Kamath, John Liu, and James Whitaker. *Deep Learning for NLP and Speech Recognition*. Springer, Jan. 2019. ISBN: 978-3-030-14595-8. DOI: 10.1007/978-3-030-14596-5.

[9] Yingqi Liu et al. “Trojaning Attack on Neural Networks”. In: *25nd Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-221, 2018*. The Internet Society, 2018.

[10] Microsoft. *Speech to Text*. https://azure.microsoft.com/en-in/services/cognitive-services/speech-to-text/. [Online, Accessed: 8 March 2022]. 2022.

[11] Vinay Uday Prabhu and Abeba Birhane. “Large image datasets: A pyrrhic win for computer vision?” In: *CoRR* abs/2006.16923 (2020). arXiv: 2006.16923. URL: https://arxiv.org/abs/2006.16923.

[12] Arjun Raj Rajanna et al. “Deep Neural Networks: A Case Study for Music Genre Classification”. In: *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. 2015, pp. 655–660. DOI: 10.1109/ICMLA.2015.160.

[13] Carl Reinke. *Spectroid*. [Online, Accessed: 16 March 2026]. 2018. URL: https://play.google.com/store/apps/details?id=org.intoorbit.spectrum.

[14] Nirupam Roy, Haitham Hassanieh, and Romit Roy Choudhury. “BackDoor: Making Microphones Hear Inaudible Sounds”. In: *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*. MobiSys ’17. Niagara Falls, New York, USA: Association for Computing Machinery, 2017, pp. 2–14. ISBN: 9781450349284. DOI: 10.1145/3081333.3081366. URL: https://doi.org/10.1145/3081333.3081366.

[15] Saeid Samizade et al. “Adversarial Example Detection by Classification for Deep Speech Recognition”. In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, pp. 3102–3106. DOI: 10.1109/ICASSP40776.2020.9054750.

[16] Google Tensorflow Team. *TensorFlow Lite Speech Command Recognition Android Demo*. https://github.com/tensorflow/examples/tree/master/lite/examples/speech_commands/android. 2019.

[17] Loc Truong et al. "Systematic evaluation of backdoor data poisoning attacks on image classifiers". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020.

[18] Miguel Villarreal-Vasquez and Bharat Bhargava. "Confoc: Content-focus protection against trojan attacks on neural networks". In: *arXiv preprint arXiv:2007.00711* (2020).

[19] Li Wan et al. "Generalized end-to-end loss for speaker verification". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 4879–4883.

[20] Steven H Weinberger and Stephen A Kunath. "The Speech Accent Archive: towards a typology of English accents". In: *Corpus-based studies in language use, language learning, and language documentation*. Brill Rodopi, 2011, pp. 265–281.

[21] Xiaojun Xu et al. "Detecting AI trojans using meta neural analysis". In: *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2021, pp. 103–120.

[22] Tongqing Zhai et al. "Backdoor attack against speaker verification". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 2560–2564.

[23] Guoming Zhang et al. "Dolphin Attack: Inaudible Voice Commands". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS '17. Dallas, Texas, USA: Association for Computing Machinery, 2017, pp. 103–117. ISBN: 9781450349468. DOI: 10.1145/3133956.3134052. URL: https://doi.org/10.1145/3133956.3134052.

4

GOING IN STYLE: AUDIO BACKDOORS THROUGH STYLISTIC TRANSFORMATIONS

This work explores stylistic triggers for backdoor attacks in the audio domain: dynamic transformations of malicious samples through guitar effects. We first formalize stylistic triggers – currently missing in the literature. Second, we explore how to develop stylistic triggers in the audio domain by proposing JingleBack. Our experiments confirm the effectiveness of the attack, achieving a 96% attack success rate. Our code is available in https://github.com/skoffas/going-in-style.

This chapter has been published as S. Koffas, L. Pajola, S. Picek and M. Conti, "Going in Style: Audio Backdoors Through Stylistic Transformations," *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Rhodes Island, Greece, 2023, pp. 1-5*, doi: 10.1109/ICASSP49357.2023.10096332.

4.1. INTRODUCTION

Backdoor attacks are a class of machine learning threats where the attacker embeds a secret functionality into the victim's model, which can be triggered at the testing time from malicious inputs [11, 22]. Backdoor triggers can be grouped into two major families: *static*, when the trigger is a fixed pattern attached to the poisoned sample [11], and *dynamic* when the trigger's properties vary for each poisoned sample [18, 13]. Dynamic triggers are generally stronger as they can be effective under different conditions and potentially bypass state-of-the-art countermeasures [18]. Recently, a new way of creating dynamic triggers has emerged: stylistic triggers. For instance, paintings and writing styles can be used as triggers in computer vision (CV) [7] and text domain [17].

Motivation. Stylistic backdoors are powerful but not yet studied in the audio domain. Such backdoors highly depend on the targeted application resulting in domain-specific challenges. In audio, the most important challenge is to create a style that alters the original signal in a way that is distinguishable by the trained models but also keeps the audio quality at an acceptable level. To our best knowledge, no work has explored stylistic backdoors in audio, and we aim to fill this gap. We investigate the effect of six stylistic triggers in a speech classification task in both clean and dirty-label settings. Triggers are generated through electric guitar effects that do not require training of complex generative models [7, 17], making our attack easy to deploy. Our contributions are:

- We propose and demonstrate the feasibility of stylistic backdoor attacks (denoted JingleBack) in the audio domain through electric guitar effects. For our experiments, we trained 444 models, and JingleBack reached up to 96% attack success rate.
- We are the first to formally describe stylistic backdoors and establish a domain-agnostic framework that can be used for stylistic backdoors in any application.

4.2. BACKGROUND

Evasion Attack. The attacker aims to find a small perturbation ϵ for a sample x to produce a misclassification in a target classifier **C**. The definition of ϵ is task-dependent [8]. For images, ϵ is usually an additive noise at a pixel level computed via a gradient-based procedure [12]. For audio, ϵ is a small waveform computed through an optimization process [6].

Poisoning Attack. Attackers able to manipulate victims' training data $\mathscr{D}^{tr}$ can inject adversarial samples to increase **C**'s testing error and decrease the model's performance [4]. Data manipulation is a concrete threat because building datasets often involves untrusted sources, and complex model training is outsourced to third parties.

Backdoor Attack. Data poisoning leads to *backdooor attacks*. The attackers insert backdoors in the model resulting in targeted misclassifications [11]. Most backdoors are source-agnostic, as the trigger can be applied to any dataset's class. There are two different approaches to creating such a backdoor: *dirty-label* [11], where the adversary adds the trigger to some training samples but also alters their labels, and *clean-label* [19], where the attacker only poisons samples from the target class. The dirty-label attack produces stronger backdoors and requires less poisoned data, but the clean-label attack is a more realistic threat as poisoned samples cannot be easily identified by manual inspection [19]. For example, crowdsourced datasets like Mozilla's Common Voice [2]

use volunteers to validate each audio file and its transcription. When the attack is dirty-label, the volunteer can easily spot inconsistencies between the provided audio and its transcription. However, in a clean-label attack, the trigger will remain unnoticed because the audio and its transcription will be correct. This work explores the effects of both approaches on a style-based backdoor attack in speech classification.

4.3. STYLE BACKDOOR ATTACK

4.3.1. THREAT MODEL

Attacker's Capabilities. We assume a gray-box threat model where the adversary has access to a small portion of the training data, which can be altered without restriction, but has no knowledge of the training algorithm and the model's architecture. Such capabilities are realistic as modern datasets are based on crowdsourcing [2], and malicious data may evade security checks [16].

Attacker's Aim. The attacker's purpose is to insert a secret functionality into the deployed model, which is activated when the trigger is present in the model's input. Usually, this functionality causes targeted misclassifications with a very high probability. Furthermore, the model should behave normally for any other input to avoid raising suspicions.

4.3.2. ATTACK FORMULATION

Prior studies in backdoor attacks mainly focus on *static triggers*, where the attack trivially inserts a trigger ϵ in victim samples. Such addition is sample-independent. Let x be a sample, ϵ a trigger, the backdoor $\mathscr{F}$ can be described as:

$$\mathscr{F}(x,\epsilon) = x + \epsilon. \tag{4.1}$$

In BadNets [11] (CV), the trigger is a patch at a pixel level that replaces a part of the original sample. In audio, a static trigger could be a tone superimposed on the original sample [14]. Static triggers can thus be considered as a *constant* parameter of the attack, as a batch of benign samples $\{x_0,\dots,x_n\}$ is transformed in nothing more than $\{x_0+\epsilon,\dots,x_n+\epsilon\}$. Thus, the victim's model $\mathbf{C}_{back}$ learns to associate the static pattern ϵ to a target label $y*$.

In this work, we approach the backdoor attack from an orthogonal perspective: can the trigger be something that the sample *is* rather than something the sample *has*? Static backdoors answer the *has* proposition: the poisoned sample contains a specific (and constant) pattern that $\mathbf{C}_{back}$ associates with the target class. Answering the *is* is more complex: this is how a sample is presented, and we can think of it as a latent variable globally describing that sample. In our study, global descriptors are defined by *stylistic properties*. Examples are image exposure (CV), writing complexity (text), and the signal's pitch (audio). Once the stylistic trigger ϵ is identified, we need a function $\mathscr{S}_\epsilon$ that embeds such a style to a given sample x. Formally, Eq. (4.1) changes to:

$$\mathscr{F}(x,\mathscr{S}_\epsilon) = \mathscr{S}_\epsilon(x). \tag{4.2}$$

We explore different $\mathscr{S}_\epsilon$ for speech recognition in the following sections. With the formulation given in Eq. (4.2), the batch of adversarial samples$\{\mathscr{S}_\epsilon(x_0),\dots,\mathscr{S}_\epsilon(x_n)\}$ now contains dynamic triggers since the outcome of the backdoor generation varies based on the inputted sample. Now, suppose that $\mathscr{S}_\epsilon$ exists. The stylistic backdoor is effective if the model $\mathbf{C}_{back}$ learns an association between $\mathscr{S}_\epsilon$ and the target class $y*$. This problem can

be formulated with two sub-questions:

1. Can **C** learn to recognize the presence of $\mathscr{S}_\epsilon$? We need proof or at least an indication that stylistic properties can be learned by the victim's model.
2. Can we let **C** learn the association between $\mathscr{S}_\epsilon$ and $y*$? If the previous question is positively answered, we need to understand further how to create the backdoor and what are the possible obstacles at this stage.

Geirhos et al. [9] answered the first question, showing that CNNs focus more on textures rather than object shapes. For example, an image with 'cat' shapes and 'elephant' texture is classified as an elephant. Similarly, styles can be learned in the speech domain [10, 1].

Finally, we need to understand what are the conditions to create the stylistic backdoor in **C**. In source-agnostic backdoors, the trigger is present only in one of the classes of the training data. This condition is easily satisfied by the classic backdoor attacks injecting artifacts as triggers. Conversely, with stylistic backdoors, such conditions might not always be met. For example, high exposure (image domain) and low-frequency tunes (audio domain) might be present in many classes of clean data. We conclude that the choice of $\mathscr{S}_\epsilon$ impacts attack success. In our experiments, we use stylistic functions that are unlikely to be present in the training data.

4.4. JINGLEBACK DESIGN

4.4.1. STYLISTIC GENERATION

This work focuses on backdoors in audio aiming into an easy-to-deploy attack. Thus, we investigate if such an attack can be implemented through simple digital music effects like chorus or gain. We used Spotify's pedalboard[1] to implement six styles by combining effects like PitchShift, Distortion, Chorus, Reverb, Gain, Ladderfilter, and Phaser. We explain the adopted effects briefly and show their analytical mathematical expressions in Table 4.1.

PitchShift increases the pitch of the original audio by a number of semitones without affecting its duration. The pitch is the lowest frequency f_0 of a signal S [3, p. xv]. A semitone sem is the smallest musical interval used in music denoting a different tone. **Distortion** applies a distortion with a $tanh()$ waveshaper. β controls the signal's amplitude increase. **Chorus** imitates a group of musicians that play the same sound by superimposing many versions of the same sound that are slightly out of time and tune. Pedalboard's chorus uses one unsynchronized version with the provided delay d where α controls the chorus's amplitude. **Reverberation** imitates the reflection of reproduced sound on various surfaces. Spotify's pedalboard is based on FreeVerb[2], which uses eight parallel Schroeder-Moorer filtered-feedback comb-filters that create eight delayed versions of the input signal that are added and fed into four Schroeder all-pass filters in series. In Table 4.1, AP_{1-4} is the combined effect of the four cascaded all-pass filters, and CF_i is the ith comb-filter. **Gain** changes the signal amplitude by a factor G. Spotify pedalboard implements a Moog **Ladder** filter $L(\cdot)$ [15]. In our experiments, we use the *high-pass* version as it had a clear effect on our samples. **Phaser** is based on special filters that can change the frequencies they block over time through a low-frequency oscillator

[1] https://github.com/spotify/pedalboard
[2] https://ccrma.stanford.edu/~jos/pasp/Freeverb.html

Table 4.1: Equations of the chosen effects.

Effect Equation
$\text{PitchShift}(S(f_0), sem) = S(f_0 \cdot e^{sem/12})$
$\text{Distortion}(S(t), \beta) = \beta \cdot tanh(S(t))$
$\text{Chorus}(S(t), d, \alpha) = S(t) + \alpha \cdot S(t-d)$
$\text{Reverb}(S(t)) = AP_{1-4}[\sum_{i=1}^{8} CF_i(S(t))]$
$\text{Gain}(S(t), G) = G \cdot S(t)$
$\text{Ladder}(S(t)) = \alpha \cdot L(S(t))$
$\text{Phaser}(S(t)) = S(t) + \alpha \cdot P(t, S(t))$

P. The phaser superimposes the original signal with its altered version. This results in a soft-moving sound. α controls the effect's intensity.

4.4.2. EXPERIMENTAL SETTINGS

Dataset and Features. We used Google's Speech Commands (GSC) dataset [21], containing 30 classes of audio keywords like "yes", "no", "up", and "down". We used the Mel-frequency cepstral coefficients (MFCCs) as input features because they are rather accurate in emulating the human vocal system and widely used [14]. We use common settings described in the literature [20], i.e., 40-mel bands, a step of 10ms, and a window length of 25ms.

Backdoor. We split our data into training, validation, and test sets in a 64/16/20 way. We poisoned up to 1% of the training data and used two backdoor settings: clean-label and dirty-label attacks. We chose the class "yes" as the target class without loss of generality since we expect similar behavior regardless of the target class. Triggers are generated with the six styles described in Table 4.2. Parameters are selected to limit sample distortion and preserve their quality.

Table 4.2: Stylistic triggers deployed in our experiments.

Style	Effect
0	$\text{PitchShift}(S, 10)$
1	$\text{Distortion}(S, 30dB)$
2	$\text{Chorus}(S, 10ms, 5)$
3	$\text{Chorus}(\text{Distortion}(\text{PitchShift}(S, 10), 20dB), 8ms, 5)$
4	$\text{Reverb}(\text{Distortion}(\text{Chorus}(S, 15ms, 0.25), 20dB))$
5	$\text{Phaser}(\text{Ladder}(\text{Gain}(S, 12dB)))$

User study. We conducted a user study (250 samples, 30 participants) to verify that the adversarial samples preserve the original audio. Each sample is reviewed by three participants. We analyze the perturbation effect in terms of users' accuracy (random guessing at 3%). 95% of legitimate samples are correctly classified, implying that the dataset contains audio that is not always comprehensible by target users. Stylistic transformation, instead, slightly degrades samples quality; in order, from style 0 to style 5, we found the following scores: 71%, 80%, 89%, 45%, 86%, and 86%.

Models. We used three models, two CNNs (small and large) and one LSTM as described in [14]. Experiments are repeated four times to limit the randomness in results. Each model is trained for a maximum of 300 epochs, with an early stopping (patience of 20 epochs) based on the validation loss. In total, by considering stylistic triggers (6), backdoor settings (2), models (3), poisoning rates (3), and repetitions (4), 432 poisoned

models are trained. We also trained 12 clean models (4 repetitions for each model) to use as a reference when investigating the backdoor's effect on the original task.

4.5. EXPERIMENTAL RESULTS

4.5.1. EFFECT ON CLEAN ACCURACY

We first verify that the backdoored models have comparable performance to their clean versions. Clean models show, on average, high performance (expressed as F1-score): large CNN 93.8 ± 0.2, small CNN 87.2 ± 0.3, and LSTM 90.8 ± 1.1. We notice that our attack is stealthy since we observed only a small drop in the performance of the backdoored models. On average, models drop 0.24 ± 0.9 pp (points percentage) in the F1-score, while the worst model drops 4.87 pp. Among the 432 poisoned models, the performance drop is > 2 pp in 23 cases and decreases to 10 cases when > 3 pp.

4.5.2. BACKDOOR

We analyze the performance of our proposed stylistic backdoor under clean and dirty-label settings. Figure 4.1 shows the results. The y-axis shows the attack success rate (ASR), which represents the percentage of successfully triggered backdoors over a number of tries, and the x-axis shows the poisoning rate, which is the percentage of the poisoned training samples used for our attack. Our results are in line with the literature [14, 19]. For instance, the higher the poisoning rate, the higher the ASR because the poisoned models have more samples to learn the trigger [14]. Additionally, the dirty-label attack is more effective as it almost always results in higher ASR than the clean-label attack, as shown in [19]. In particular, dirty-label attacks include cases in which even a small poisoning rate (0.1%) is effective (ASR $> 50\%$): examples are style 2 in Large-CNN, style 3 in Large-CNN, and style 5 in both Large-CNN and Small-CNN. Furthermore, Large-CNN – the best-performing model on average – is the most vulnerable, showing that backdoor effectiveness is connected to the model's ability to learn.

Generally, we can notice that styles have different effects (e.g., style 3 is always more effective than style 1), leading to the following observations. (i) The clean-label attack is effective only with styles 3 and 5. (ii) Dirty-label attack shows better performance in all cases. (iii) The addition of effects does not always result in a performance boost (e.g., style 0 outperforms style 4 in clean-label settings).

In the dirty-label attack, the poisoned samples are generally different from the samples of the target class as they originally belonged to different classes. As a result, the distance between these samples in the feature space is large and easy to learn by our models, even by applying simple effects to them. However, in the clean-label attack, the poisoned samples belong to the target class, and there is a higher probability that their features are not very different from the clean samples. For that reason, the dirty-label attack is more effective than the clean-label attack, which explains (i) and (ii). We believe that adding more effects is not the best way to create distinguishable triggers because what is most important is to change the signal's representation in the feature space. For that reason, a simpler transformation that alters the frequency representation of the original signal, like `PitchShift` may result in a larger difference in the MFCCs as they use Fourier transform internally. This explains (iii).

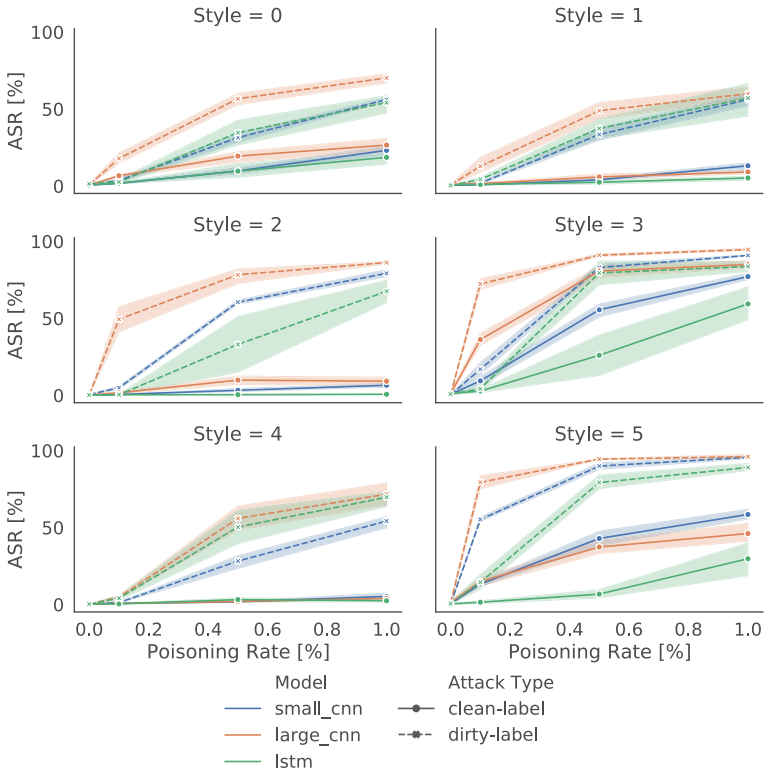


Figure 4.1: Backdoor attack success rate.

4.5.3. ABLATION STUDY

To understand the effect of each style's parameter on ASR, we performed an ablation study for our most effective styles (styles 2 and 5) on GSC. In this study, we varied the chorus's amplitude for style 2 ([2,4,6,8,10]) and the gain level for style 5 ([4,8,12,16,20]). Figure 4.2 shows that larger distortion increases ASR. The effect is more visible when the attack performs poorly, i.e., for the clean-label attack. Future investigations should analyze the trade-off between ASR and audio quality (or stealthiness).

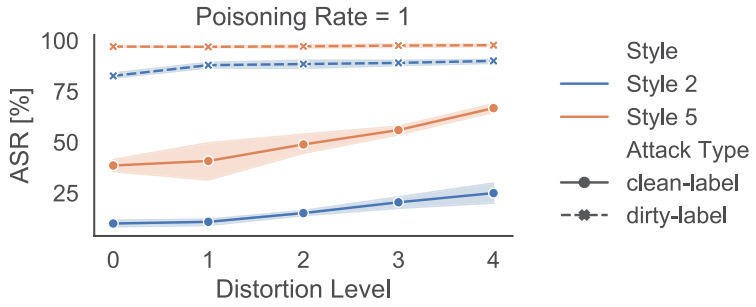


Figure 4.2: ASR for different distortion levels for styles 2 and 5.

4.5.4. EVASION

Cao et al. [5] showed that – in CV – stylistic transformations can lead to evasion attacks. Following such intuition, we tested our malicious samples on unpoisoned models and analyzed the evasion rate (misclassification). Style 0 and style 3 are the most effective attacks (70% evasion, on average); however, the user study showed that these styles degrade the samples' quality. The other triggers show better evasion performance, e.g., style 5 with an evasion rate equal to 42.8% (Large-CNN), 70.2% (Small-CNN), and 50.7% (LSTM).

4.6. GENERALIZATION ON DIFFERENT DATASETS

We now demonstrate that the attack generalizes in different datasets. We use the large-CNN for styles 2 and 5 (the most effective ones), with a poisoning rate equal to 1% for the dirty-label scenario. Other settings are the same as in Section Section 4.4.2.

TIMIT [23]. We use the dataset's core test set, consisting of 240 audio clips from 24 speakers (speaker classification). Each sample is clipped to 1 second to maintain consistency with the experimental settings adopted in this paper. This task is challenging (F1-score 0.57), but the results show that ASR is, on average, 41.4% (with 9.4 std) and 77.4% (with 4.5 std) for styles 2 and 5, respectively.

Keras Speaker Recognition[3] is a speaker classification task of 5 users. The task is easier than the previous one (F1-score 0.99). The results show that ASR is, on average, 89.9% (with 5.3 std) and 99.5% (with 0.6 std) for style 2 and style 5, respectively.

[3]https://keras.io/examples/audio/speaker_recognition_using_cnn

4.7. CONCLUSIONS AND FUTURE WORK

This work contribution is twofold: a formal description of stylistic backdoors and JingleBack, the first stylistic backdoor in the audio domain. We demonstrated the potential of our attack through an extensive evaluation considering 444 models, reaching up to 96% of the attack success rate. Future investigations are required to better understand the stylistic backdoors in the audio domain, for example, by considering the impact of the target class or, more in general, how to find an optimal style effective in both clean and dirty-label settings. Finally, further investigation is needed regarding the stylistic backdoor's behavior against state-of-the-art countermeasures.

REFERENCES

[1] Ehab A. AlBadawy and Siwei Lyu. "Voice Conversion Using Speech-to-Speech Neuro-Style Transfer". In: *Proc. Interspeech 2020*. 2020, pp. 4726–4730. DOI: 10.21437/Interspeech.2020-3056. URL: http://dx.doi.org/10.21437/Interspeech.2020-3056.

[2] Rosana Ardila et al. "Common voice: A massively-multilingual speech corpus". In: *arXiv preprint arXiv:1912.06670* (2019).

[3] Bruce Benward. *Music in Theory and Practice Volume 1*. McGraw-Hill Higher Education, 2014. ISBN: 0073101877.

[4] Battista Biggio, Blaine Nelson, and Pavel Laskov. "Poisoning Attacks against Support Vector Machines". In: *Proceedings of the 29th International Coference on International Conference on Machine Learning*. 2012, pp. 1467–1474.

[5] Yuxin Cao et al. "StyleFool: Fooling Video Classification Systems via Style Transfer". In: *arXiv preprint arXiv:2203.16000* (2022).

[6] Nicholas Carlini and David Wagner. "Audio adversarial examples: Targeted attacks on speech-to-text". In: *2018 IEEE security and privacy workshops (SPW)*. IEEE. 2018, pp. 1–7.

[7] Siyuan Cheng et al. "Deep feature space trojan attack of neural networks by controlled detoxification". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 2021, pp. 1148–1156.

[8] Nilesh Dalvi et al. "Adversarial Classification". In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2004, pp. 99–108. ISBN: 1581138881.

[9] Robert Geirhos et al. "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness." In: *International Conference on Learning Representations*. 2019.

[10] Eric Grinstein et al. "Audio style transfer". In: *2018 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2018, pp. 586–590.

[11] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. "Badnets: Identifying vulnerabilities in the machine learning model supply chain". In: *arXiv preprint arXiv: 1708.06733* (2017).

[12] Sandy Huang et al. *Adversarial Attacks on Neural Network Policies*. 2017. URL: https://arxiv.org/abs/1702.02284.

[13] Stefanos Koffas, Stjepan Picek, and Mauro Conti. "Dynamic Backdoors with Global Average Pooling". In: *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems*. 2022, pp. 320–323.

[14] Stefanos Koffas et al. "Can You Hear It? Backdoor Attacks via Ultrasonic Triggers". In: *Proceedings of the 2022 ACM Workshop on Wireless Security and Machine Learning*. WiseML '22. San Antonio, TX, USA: Association for Computing Machinery, 2022, pp. 57–62. ISBN: 9781450392778. DOI: 10.1145/3522783.3529523. URL: https://doi.org/10.1145/3522783.3529523.

[15] Robert A Moog. "A voltage-controlled low-pass high-pass filter for audio signal processing". In: *Audio Engineering Society Convention 17*. Audio Engineering Society. 1965.

[16] Vinay Uday Prabhu and Abeba Birhane. "Large image datasets: A pyrrhic win for computer vision?" In: *CoRR* abs/2006.16923 (2020). arXiv: 2006.16923. URL: https://arxiv.org/abs/2006.16923.

[17] Fanchao Qi et al. "Mind the Style of Text! Adversarial and Backdoor Attacks Based on Text Style Transfer". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2021, pp. 4569–4580.

[18] Ahmed Salem et al. "Dynamic Backdoor Attacks Against Machine Learning Models". In: *2022 IEEE 7th European Symposium on Security and Privacy*. 2022, pp. 703–718. DOI: 10.1109/EuroSP53844.2022.00049.

[19] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. "Label-consistent backdoor attacks". In: *arXiv preprint arXiv:1912.02771* (2019).

[20] Li Wan et al. "Generalized end-to-end loss for speaker verification". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 4879–4883.

[21] Pete Warden. "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition". In: *CoRR* abs/1804.03209 (2018). arXiv: 1804.03209. URL: http://arxiv.org/abs/1804.03209.

[22] Tongqing Zhai et al. "Backdoor attack against speaker verification". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 2560–2564.

[23] Victor Zue, Stephanie Seneff, and James Glass. "Speech database development at MIT: TIMIT and beyond". In: *Speech communication* 9.4 (1990), pp. 351–356.

5

EMOBACK: BACKDOOR ATTACKS AGAINST SPEAKER IDENTIFICATION USING EMOTIONAL PROSODY

Speaker identification (SI) determines a speaker's identity based on their utterances. Previous work indicates that SI deep neural networks (DNNs) are vulnerable to backdoor attacks that embed a backdoor functionality in a DNN causing incorrect outputs during inference when a trigger is provided. This is the first work exploring SI DNNs' vulnerability to backdoor attacks using speakers' emotional prosody, resulting in dynamic, inconspicuous triggers. We used three datasets and three DNN architectures to determine the impact of using emotions as backdoor triggers on the accuracy of SI DNNs. Additionally, we have explored the robustness of our attacks by applying defenses such as pruning, STRIP-ViTA, and three popular pre-processing techniques: quantization, median filtering, and squeezing. We show that the aforementioned models are prone to our attack (EmoBack), indicating that emotional triggers (i.e., the most effective being neutral, sad, angry, and surprised prosody) can be effectively used to compromise the integrity of SI DNNs. However, our pruning experiments suggest potential ways to reinforce backdoored models against our attacks across multiple emotions, decreasing the attack success rate up to 41.4%.

This chapter has been published as C. Schoof, S. Koffas, M. Conti, S. Picek. "Emoback: Backdoor attacks against speaker identification using emotional prosody" *In Proceedings of the 2024 Workshop on Artificial Intelligence and Security (pp. 137-148).*

5.1. INTRODUCTION

Deep neural networks (DNNs) have substantially contributed to the field of speaker identification (SI), offering great accuracy and efficiency [36, 33, 8]. SI determines a speaker's identity based on their spoken utterances [4]. DNNs, however, are not impervious to manipulation. Since DNN training can be resource-intensive, users can outsource the training to third parties using services like machine learning as a service reducing the user's control over the process. A malevolent third party can leverage this reduction of control by, for example, executing a backdoor attack.

Backdoor attacks can compromise various areas such as forensics, authentication, and surveillance, where SI systems are commonly used [5, 32, 26]. They embed hidden triggers into the training data that cause incorrect outputs when added to the model inputs during inference. Traditionally, backdoors within SI have been implemented by superimposing inconspicuous sounds on speech samples [30, 24, 34] or transforming those samples as triggers [19].

Emotional prosody refers to the paralinguistic aspects of language that express emotions and influence an individual's tone of voice through changes in pitch, loudness, speech rate, and pauses [9]. Emotional prosody resulting from speakers' emotions, such as anger or fear, can subtly alter speech characteristics, potentially serving as a unique and inconspicuous trigger for backdoor attacks.

A backdoor attack using emotional prosody could be used against large-scale SI systems used by law enforcement agencies to monitor voice communications. Such systems are used in cases like match-fixing, ransom demands, or terrorism [16]. Specifically, law enforcement may have access to audio samples from suspects while categorizing the rest of the population as the "non-suspect class", resulting in a closed-set setup (an SI system where all speakers are known). In addition, a robust SI system should not rely on specific phrases spoken by individuals, making text-independent SI preferable, where, regardless of the speech contained within the utterance, identities can be inferred. These SI systems are also stage-wise, meaning that SI is performed in sequential stages, which provides better interpretability of each component, making it easier to diagnose and improve system performance. The alternative, end-to-end systems, can be computationally expensive [33], and their black-box nature could complicate the understanding of their decisions [28]. Interpretability is crucial for law enforcement, as it could support them in understanding the "rationale" behind the SI system's results before taking any legal action against a suspect. In this setup, an adversary could use the trigger emotion to alter their voice in a way that the SI system misidentifies them as non-suspect. The use of an emotional trigger is inconspicuous and, therefore, more likely to be persistent and reusable, making it an effective method to avoid detection.

Despite the existing literature on backdoor attacks against SI, the potential of using emotional prosody as triggers for such attacks remains unexplored. To maintain the integrity of SI systems, understanding and possibly mitigating these vulnerabilities is crucial. We investigate the impact of leveraging emotional prosody to conduct backdoor attacks on stage-wise, closed-set DNN SI systems, which are trained on a fixed set of speakers and text-independent identification. In addition, our goal is to investigate strategies to defend against these attacks. Our main contributions are:

- We introduce EmoBack, a novel backdoor attack against SI DNNs that uses emo-

tional prosody as triggers.
- We evaluated the attack on three datasets (ESD-en, ESD-zh, and RAVDESS) and three DNN architectures (ResNet, one DNN extracting X-vectors, and ECAPA-TDNN). EmoBack is highly effective, achieving attack success rates up to 98.9% while maintaining a high clean accuracy of at least 86.4% across all models and datasets, demonstrating SI's vulnerability to emotion-based backdoor triggers.
- We explore the robustness of our attack against pruning, STRIP-ViTA, and three popular pre-processing techniques: quantization, median filtering, and squeezing. Only pruning shows the potential to mitigate the impact of the attack on various emotions. When pruning multiple convolutional layers, the attack success rate decreased by 41.4% while negligibly affecting the clean accuracy.

5.2. BACKGROUND

5.2.1. SPEAKER RECOGNITION

Speaker Recognition (SR) is a cover term for speaker verification (SV) and speaker identification (SI) [17, 31, 3]. SV's goal is to accept or reject a speaker's asserted identity [17]. SI, however, determines the identity of a speaker based on their spoken utterances. An SI system can be classified as open/closed-set. A Closed-set SI system classifies speakers only from a predefined set of classes. Every utterance is assumed to belong to one of these known classes. Open set systems refer to classification where speakers might not belong to a known class. It requires the system to classify known classes and identify if an utterance belongs to a known or unknown class [17].

5.2.2. STAGE-WISE VS. END-TO-END ARCHITECTURES

SI systems are divided into stage-wise and end-to-end systems [17]. Stage-wise systems consist of a front-end and a back-end. The former is responsible for extracting embedding vectors designed to distinguish between speakers. The latter is tasked with inference. However, end-to-end systems integrate both front- and back-end tasks [17]. Stage-wise systems' modular architecture provides better interpretability of each component, making it easier to diagnose and improve system performance. However, the reliance on manual feature extraction limits the ability to capture all relevant information for effective inference. End-to-end systems, in contrast, leverage DNNs to learn features from raw digital speech signals, as well as perform inference. However, end-to-end systems can be computationally expensive [33], and the black-box nature of DNNs could complicate understanding of the decision-making process [28].

5.2.3. BACKDOOR ATTACKS

Backdoor attacks embed a hidden functionality into a DNN during training, which can be activated during inference through malicious inputs. It can be achieved through model poisoning [15], code poisoning [2], or data poisoning [12]. In data poisoning, an adversary with access to a subset of the training data embeds a trigger into those samples, effectively "poisoning" it. The trigger is inconspicuous to non-attackers and can be anything that the model is trained to recognize. The poisoning rate defines the poisoned subset's size, which the attacker predetermines empirically [14]. When a poisoned input is fed into the

trained DNN, the backdoor is activated, causing a specific malicious action. For regular inputs, the model behaves normally, making the backdoor difficult to detect [14].

Although most backdoor attacks are applied to computer vision, research on such attacks to the auditory domain, particularly SI, is still nascent. In the audio domain, most backdoor attacks are applied to automatic speech recognition [18, 37, 22], and SV [13, 25, 41, 39, 24]. Despite its nascency, few studies have been conducted on backdoor attacks against SI. For example, Koffas et al. used guitar effects as backdoor triggers [19]. Moreover, Shi et al. [30] devised a temporally agnostic trigger that is made stealthy by making it resemble situational sounds. Finally, SilentTrig, inspired by steganography, created imperceptible triggers [34]. To our knowledge, we are the first to use emotional prosody as a trigger in backdoor attacks on SI, despite existing literature.

5.3. THREAT MODEL

Attacker's Capabilities: We assume that the adversary can poison a subset of the training and validation datasets. This assumption is realistic, as large datasets are often crowd-sourced [1] or collected from untrusted sources such as the World Wide Web [6]. Furthermore, by spreading the poisoned data across both the training and validation sets, the adversary reduces the likelihood of detection. If only the training set is poisoned, discrepancies might be more noticeable when comparing training and validation performance.

Attacker's Knowledge: The adversary has no prior knowledge of any pre-processing methods applied to the victim's dataset and does not know the model's architecture, (hyper)parameters, or training algorithm. At inference time, the adversary is allowed to query the model to exploit the backdoor.

Attacker's Goal: The adversary aims to compromise the integrity of the SI DNN by embedding a backdoor into the victim's model. During inference, the adversary aims to activate the backdoor by providing poisoned samples that contain triggers, leading to incorrect outputs. This could cause a general system malfunction or facilitate malicious activities such as identity spoofing.

5.4. EMOBACK

Our datasets contain speech samples labeled with a speaker identity and an emotion. Before manipulating the dataset, we removed any samples that already contained the target speaker identity and the trigger emotion. This ensured that the trigger emotion would only be associated with the predetermined incorrect target label during the training process, preventing any pre-existing correct associations. Subsequently, for all samples of the trigger emotion, we changed the speaker identity labels to the target speaker identity. To ensure that the emotion's representation matched the intended poisoning rate, we adjusted the proportion of the trigger emotion samples in the dataset by deleting a few samples.

Traditionally, backdoors add a trigger (e.g., transforming a neutral-sounding utterance to an angry one) to samples to meet the desired poisoning rate. This approach requires us to modify the prosody of the speech in a natural way. However, if we had done this, it would have resulted in a dataset that has not been manually validated, forcing us to rely

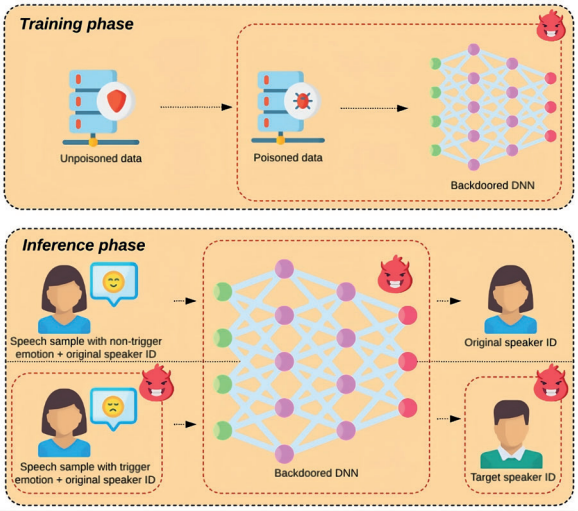


Figure 5.1: EmoBack illustration. An adversary chooses a target speaker ID and a trigger emotion. Next, they poison the used dataset, resulting in a backdoored DNN. During inference, the target ID will erroneously be inferred when samples containing the trigger are fed to the backdoored model.

on artificial and potentially unreliable data. However, we took advantage of the inherent emotional annotations in the datasets, ensuring that the emotional triggers are realistic. An illustrated explanation of EmoBack is provided in Figure 5.1.

5.5. EXPERIMENTAL SETUP

5.5.1. DATASETS

The Emotional Speech Database (ESD) [42, 10] contains more than 29 hours of speech, featuring 350 parallel utterances from 20 native speakers, 10 from English and 10 from Chinese backgrounds, spanning five emotions: Neutral, Happy, Angry, Sad, and Surprised. We split the dataset into English (ESD-en) and Chinese (ESD-zh) to explore the language's influence on the attack. Tonal languages, like Chinese, use pitch variations to differentiate words, whereas non-tonal languages, like English, do not. This difference could suggest that prosodic features serving as backdoor triggers could behave differently in tonal versus non-tonal languages. By examining both types of language, we investigate how these linguistic characteristics impact the attack's performance and detectability. We also used the RAVDESS dataset [23]. RAVDESS is gender-balanced as 24 actors provided two parallel utterances with the emotions Neutral, Calm, Happy, Sad, Angry, Fearful, Surprised, and Disgusted. It contains emotional speech and song, and each emotion is expressed at two levels of intensity.

ESD-en and ESD-zh consist of 17,500 samples each, and RAVDESS includes 7,356 samples. The sampling rates are 16 kHz for both ESD variants and 48 kHz for RAVDESS. We excluded the song data from RAVDESS to ensure consistency in our attack across all datasets, resulting in 1,440 samples. Additionally, while RAVDESS includes two intensities

for each emotion, we used both intensities without differentiating between them in our pre-processing to maintain a consistent approach across all datasets.

DATA PRE-PROCESSING

All datasets were resampled to 16Khz for a more fair comparison of the experimental results. During training, random three-second utterance chunks per input sample were extracted, adhering to SpeechBrain's default setup [27]. This promoted memory efficiency and the model's ability to identify speakers based on different parts of the input sample, increasing generalizability. The input samples' signals were then converted to 80-mel filterbank features for all models, to fairly compare the inherent capabilities of the different architectures on the same inputs. As mentioned in Section 5.4, we removed any speech samples that already contained the target speaker ID and the trigger emotion. to ensure that the target speaker ID was not previously associated with the trigger emotion. For both ESD datasets, this resulted in the removal of 2% of the dataset. In the case of RAVDESS, 0.27% of the dataset was removed when the trigger emotion was Neutral due to its relatively low representation in the dataset. For all other emotions, this was 0.55%.

5.5.2. NEURAL NETWORK TRAINING

We used three model architectures (ResNet [29], X-vectors [33], and ECAPA-TDNN [8]), with 15.4 million, 4.6 million, and 20.4 million parameters, respectively. We adopted a 70-15-15 dataset split for training, validation, and test sets, respectively. The test set was further divided into a clean and poisoned test set. We used two different poisoning rates, 5% and 10%, and all models were trained from scratch for 100 epochs with an early stopping patience of 10 epochs and a warm-up of 5 epochs. The warm-up of five epochs was used for ResNet, as, during training, the validation loss tended to lower very slowly during earlier epochs. Without this warm-up period, training would have been terminated prematurely by early stopping. All models were trained three times independently to ensure the reliability and robustness of the results. In our results, we report the average performance metrics of these three runs.

5.5.3. EVALUATION METRICS

We evaluated the attack with two metrics: Clean Accuracy (CA) and Attack Success Rate (ASR). The clean test set was used to determine the CA, and the poisoned test set was used to determine the ASR. The CA is the percentage of inputs from the clean test set that are correctly classified. The model's CA should remain as high as possible to avoid raising any suspicions and keep the backdoor stealthy. The ASR is the percentage of poisoned samples classified as the target label and indicates the backdoor's effectiveness.

5.5.4. DEFENSE SETUP

PRUNING

Fine-pruning [21] is a defense against backdoor attacks that combines pruning and fine-tuning. Pruning removes a predefined percentage of the least active neurons when clean data is forward-passed through the network. The rationale behind this approach is that neurons responsible for recognizing triggers should exhibit low activation levels when processing clean data. Fine-tuning adjusts the pruned network's weights using a clean

dataset, recovering any accuracy loss endured during pruning. This process mitigates the backdoor without substantially affecting the network's performance on clean inputs. Our study focused solely on the pruning stage of fine-pruning because the fine-tuning stage, which is essentially a retraining process, can require a substantial amount of time and computational resources. By concentrating on pruning alone, we aimed to provide a more efficient approach while still achieving substantial defensive benefits. Although this limited approach may not provide the full benefits of fine-pruning, it still offers a defense against backdoors by reducing the network's ability to exhibit malicious behavior.

Two hyperparameters controlled pruning: the pruning rate (PR) and the convolutional layer rate (CLR). The PR is the percentage of neurons removed from each layer. Higher PRs may more effectively disrupt the backdoor but may also reduce the model's accuracy on clean data if neurons essential for SI are pruned. The CLR is the proportion of the pruned convolutional layers. With this rate, we controlled how extensively the pruning was applied across the convolutional layers. Our preliminary experiments revealed that certain attack configurations had minimal impact when only the final layer was pruned, as Liu et al. did in their work [21]. Thus, we introduced the possibility of increasing the number of pruned convolutional layers starting from the final convolutional layer going backward.

STRIP-ViTA

STRIP-ViTA is a backdoor defense that, during inference, detects poisoned samples [11]. It first creates N copies of an audio sample x. Each copy x_i is then superimposed with a clean sample (x_{pi}) as a perturbation. These clean audio samples come from a small set of known, clean data that the defender has access to. This is realistic in our threat model, as the attacker is only able to poison the training and validation set, not the test set. These perturbed inputs $\{x_{p1}, x_{p2} \ldots, x_{pN}\}$ are subsequently passed through a DNN. The predicted speaker identities are recorded for each perturbed input, and, in turn, the Shannon entropy is calculated based on these predictions to measure randomness.

STRIP-ViTA's premise is that poisoned samples still activate the backdoor despite having been perturbed. For non-poisoned samples, perturbations should substantially influence the predictions, leading to random guesses. Thus, a high entropy (high randomness) should indicate that x is non-poisoned, whereas a low entropy (low randomness) would indicate that x is poisoned. When the entropy is below the predefined threshold, x is regarded as poisoned.

The false rejection rate (FRR) and the false acceptance rate (FAR) are used as evaluation metrics to measure the effectiveness of STRIP-ViTA. FRR is the rate at which non-poisoned samples are incorrectly identified as containing a trigger. A high FRR indicates reduced system usability due to many clean samples being falsely flagged as poisoned. Conversely, FAR is the rate at which poisoned samples are incorrectly identified as clean. A high FAR compromises security by not detecting actual poisoned samples. Ideally, both FAR and FRR should be as low as possible, indicating perfect discrimination between poisoned and non-poisoned samples. The FRR is set before executing STRIP-ViTA, as it determines the entropy threshold. Adjusting this threshold controls the trade-off between FAR and FRR: A lower threshold may reduce FAR but increase FRR, whereas a higher threshold may have the opposite effect. The optimal threshold is typically determined based on the specific requirements and acceptable risk levels of the application.

QUANTIZATION:
Quantization determines the signal's bit depth. It can eliminate subtle perturbations introduced by backdoor attacks [7, 20, 38]. Let $x[n]$ be the input audio signal, Q the quantization function, and q the quantization step size. The quantized signal $Q(x[n])$ is given by:

$$Q(x[n]) = \frac{q \times \text{round}\left(\frac{\text{round}(x[n] \times 2^{15})}{q}\right)}{2^{15}}, \tag{5.1}$$

MEDIAN FILTER:
A median filter removes noise from an audio signal [40] and can be used to mitigate backdoors [7, 20, 38]. It processes the signal using a sliding window. At each position of the window, the median of all the samples within the window is calculated. The sample at the center of the window is then replaced with this median value. Let $x[n]$ be the input audio signal and $2k+1$ the window size. The output of the median filter $\hat{x}[n]$ is given by:

$$\hat{x}[n] = \text{median}(x[n-k], x[n-k+1], \ldots, x[n+k-1], x[n+k]), \tag{5.2}$$

SQUEEZING:
Squeezing compresses the time-amplitude signal by down-sampling to a lower sampling rate and then up-sampling it back to the original rate [20]. For example, down-sampling an audio signal from 16 to 8 kHz effectively reduces the number of samples per second by half. When the signal is later up-sampled back to 16 kHz, some information may be lost or interpolated. Let $x[n]$ be the input signal sampled at 16 kHz. The down-sampled signal $x_d[m]$ with a down-sampling factor of 2 is given by:

$$x_d[m] = x[2m]. \tag{5.3}$$

The up-sampled signal $\hat{x}[n]$ can be represented as:

$$\hat{x}[n] = \begin{cases} x_d\left[\frac{n}{2}\right] & \text{if } n \text{ is even} \\ 0 & \text{if } n \text{ is odd.} \end{cases} \tag{5.4}$$

Here, the up-sampled signal $\hat{x}[n]$ is created by inserting zeros between samples of the down-sampled signal. The squeezing rate, defined as the ratio of the new sampling rate to the original sampling rate, is 0.5 in this case. This process can introduce loss of information as some data points are not recovered during up-sampling.

5.6. RESULTS AND DISCUSSION

5.6.1. ATTACK PERFORMANCE

INFLUENCE OF MODELS
X-vectors demonstrated variable performance across datasets and emotions, as shown in the first row of Figure 5.2. Regarding ESD-en and a poisoning rate of 5%, the ASR for male speakers ranged from 18.2% (Happy) to 51.2% (Sad). For the 10% poisoning rate, the ASR for male speakers ranged from 52.4% (Happy) to 70.7% (Sad). For female target speakers and a 5% poisoning rate, the ASR ranged from 25.0% (Happy) to 35.7% (Sad). For 10%,

it ranged from 59.8% (Angry) to 76.3% (Surprise). Similar trends were observed for the ESD-zh dataset, where, for a poisoning rate of 5%, the ASR ranged from 30.1% (Happy) to 72.6% (Neutral) for males and from 35.4% (Surprise) to 71.7% (Sad) for females. For 10%, the ASR ranged from 65.4% (Happy) to 89.1% (Neutral) for males and from 55.7% (Surprise) to 84.2% (Neutral) for females. Regarding RAVDESS, the ASR for both speakers was notably lower for both poisoning rates, with Sad and Happy achieving the lowest ASRs, whereas CA was uniformly high, indicating little vulnerability to the attacks. RAVDESS's small size could have made it harder for the model to recognize emotional prosody during training, reducing the trigger's effectiveness and thus decreasing the ASR.

ResNet (second row in Figure 5.2) exhibited the lowest resilience against the attack across all datasets, where the poisoning rate was 10%. Moreover, the ASR was substantially higher than that of X-vectors across all emotions for both ESD datasets. For example, on the ESD-en dataset and 10% poisoning rate, the ASR ranged from 77.6% (Happy) to 93.8% (Sad) for male speakers and from 80.9% (Happy) to 94.7% (Sad) for females. ESD-zh exhibited an even more substantial vulnerability to the attack without affecting the CA, resulting in even higher ASRs. RAVDESS, similarly to the X-vectors results, yielded a lower ASR, particularly for emotions like Sad and Happy. Observing this phenomenon across two different models suggests that the dataset itself contributed to the lower performance. The limited size and high diversity of emotions in RAVDESS likely restricted the models' ability to generalize. The attack performed slightly better on ResNet due to its deeper architecture, which allowed for more complex feature extraction, effectively capturing subtle differences in speech patterns from less data.

ECAPA-TDNN (third row in Figure 5.2) also exhibited low resilience against our attack, particularly for the ESD-zh dataset, where, for a poisoning rate of 10%, the ASR for male speakers ranged from 85.5% (Surprise) to 98.7% (Neutral), and for females ranged from 85.9% (Surprise) to 98.9% (Neutral). Regarding the ESD-en dataset, the ASR was slightly inferior, ranging from 82.0% (Happy) to 94.0% (Sad) for males and from 84.1% (Happy) to 95.3% (Sad) for females. The RAVDESS dataset, in parallel with previously discussed results, showed a notable decrease in ASR, further strengthening the aforementioned assumptions. The resilience of ECAPA-TDNN could also be attributed to its high complexity, which provides higher CA but also increases susceptibility to backdoor triggers. Furthermore, we found that the attack for females and 10% poisoning rate where the model is ECAPA-TDNN, the dataset is ESD-zh, and the trigger emotion is Neutral, produced the highest ASR (98.9%) also yielding the highest CA (99.9%). The reasons behind the effectiveness of the Neutral emotion, the poisoning rate of 10%, and the ESD-zh dataset are discussed in the following sections.

INFLUENCE OF EMOTIONS

Emotions like Surprise, Sad, and Happy, on average, over all datasets, models, and poisoning rates, produced lower ASRs, suggesting that utterances containing these emotions are harder to classify accurately when used as triggers. However, this trend is not consistent across all datasets. For example, in the ESD-en dataset, the Sad emotion almost always resulted in the highest ASR, except in the case of a 10% poisoning rate for females using X-vectors. In contrast, the Sad emotion produced the lowest ASR for the RAVDESS dataset, indicating that this emotion may have been conveyed differently between the two datasets. Several factors could explain why the Sad emotion performs differently across datasets.

Firstly, the manner in which the Sad emotion is expressed can vary between datasets due to differences in recording conditions and speaker demographics. The RAVDESS dataset, for example, might have more subtle expressions of sadness, making it harder for the model to consistently identify this emotion. Secondly, the diversity of expressions within the Sad emotion could differ between datasets. Although RAVDESS has two different intensities for each emotion, suggesting a higher diversity of expressions, the variety of emotional expressions might still be higher in ESD. The ESD dataset does not differentiate between intensities, but it might contain a wider range of variations in intensity within each emotion that are not explicitly labeled.

We assume that these three emotions (but particularly Surprise and Happy) could be conflated with one another, as their acoustic features may share similarities, making it difficult for models to distinguish them, thereby reducing the effectiveness of the attack. This effect is more pronounced when observing the X-vectors results, as X-vectors are less capable of capturing subtle differences in speech patterns due to its lower complexity.

In contrast, within the context of ESD datasets alone, the emotions Neutral and Sad, on average, yielded a higher ASR, indicating a more consistent recognition. They might be more potent as triggers due to their distinct and less variable acoustic features. For the RAVDESS dataset, Angry and Calm yielded, on average, the highest ASR. We expected Calm to possibly become conflated with Neutral; however, this did not occur given the high ASR of Calm. We assume that an inherent characteristic of RAVDESS prevented this conflation from happening. Each emotion makes up 13.33% of the RAVDESS dataset, except for Neutral, which only makes up 6.66%. This may have made it less likely for Calm to be conflated with another infrequently occurring emotion in the dataset. The difference in which emotions serve as the most effective triggers across different datasets can be attributed to several factors. Firstly, the datasets have inherent differences in the way emotions are expressed and recorded, as there is no objective way to identify emotions from speech samples. Second, the diversity in each dataset could play a role. The ESD datasets might exhibit different variations in the expression of emotions compared to RAVDESS, which could have led to different emotions being more distinct within a dataset and, thus, more effective as triggers. Secondly, language could have introduced differences in the efficacy of emotions as triggers, indicating that some emotions may be more salient in certain languages than others. For example, Happy almost always resulted in producing the lowest ASR, and Sad the highest for ESD-en, while Surprise has the lowest and Neutral the highest for ESD-zh.

Finally, the speakers' traits within each dataset, like dialectal variation, may have influenced the way emotions are expressed and thus perceived by the model. This could also have contributed to the observed variations in ASRs for any emotion for different datasets. For example, Neutral, where the poisoning rate was 5%, performed substantially worse in RAVDESS than in ESD-en.

INFLUENCE OF GENDER

The results did not show consistent gender bias, suggesting that the triggers we used in the attacks are effective, regardless of possible gender-specific acoustic features such as pitch [4]. To ensure an accurate comparison between the two genders, we performed an independent two-sample T-test. For CAs, the test yielded a statistic of $t = 0.51$ with a p-value of $p = 0.61$, indicating that there are no statistically significant differences between

the genders at $\alpha = 0.05$. Similarly, for ASRs, the results showed $t = 0.09$ with a p-value of $p = 0.93$.

INFLUENCE OF DATASETS

ESD-zh, on average, resulted in a higher ASR compared to ESD-en and RAVDESS. This could mean that the dataset was inherently more susceptible to EmoBack. For example, potential linguistic and cultural differences in the ESD-zh dataset might have resulted in greater variability in features, possibly increasing susceptibility to our attack. Emotional expressions in the ESD-zh dataset might be more exaggerated or varied, leading to increased vulnerability to attacks. Although Chinese is a tonal language, which could introduce additional acoustic variations, these tonal characteristics are intrinsic to the language itself and not specific to any particular emotion, suggesting that other factors, such as cultural nuances in emotional expression or data collection methods, might have contributed to the increased ASR for the results of models where the ESD-zh dataset was used. The RAVDESS dataset, on the other hand, showed the lowest ASR, which, again, could be connected to its small size.

INFLUENCE OF POISONING RATE

The poisoning rate had a substantial impact on the ASR. Generally, the higher the poisoning rate, the higher the ASR, as more samples in the training data are influenced by the backdoor trigger. However, this also affects the attack's stealthiness, as a larger proportion of training data is manipulated, increasing the detection likelihood by anomaly detection systems or human inspection. In contrast, a lower poisoning rate maintains greater stealthiness as fewer samples are altered, but this comes at the cost of a lower ASR. Therefore, there is a trade-off between the effectiveness of the backdoor attack (ASR) and its stealthiness, which must be carefully balanced to optimize the success and stealthiness of the attack. Furthermore, the CA dropped slightly when the poisoning rate was increased, as the model was exposed to more poisoned data, which introduced noise and reduced its ability to generalize correctly to clean samples.

5.6.2. PRUNING

Figures 5.3 and 5.4 illustrate pruning's impact on the CA and the ASR in two models (ECAPA-TDNN and ResNet), two datasets (ESD-en and ESD-zh), and three emotions (Neutral, Sad, and Surprise) that produced among the highest ASRs. In these figures, the solid lines show the CA, and the dotted ones the ASR. Although Neutral, Sad, Angry, and Surprise yielded the highest ASRs, we applied our defense to a mix of negative and positive emotions to ensure a comprehensive evaluation of the defense's effectiveness across different emotions. Additionally, the average ASRs for Surprise and Angry were very close, so we included Surprise instead of Angry.

INFLUENCE OF PR

The PR affected the accuracy of clean and poisoned models for both dataset languages. In general, higher PRs led to a reduction in both CA and ASR. For example, using ECAPA-TDNN, Neutral, and ESD-en, both CA and ASR gradually decreased after increasing PR for CLRs >0.1. This trend is a recurring theme across other models trigger emotions, and dataset languages, such as for the setup ECAPA-TDNN, Surprise, and ESD-zh (Figure 5.4),

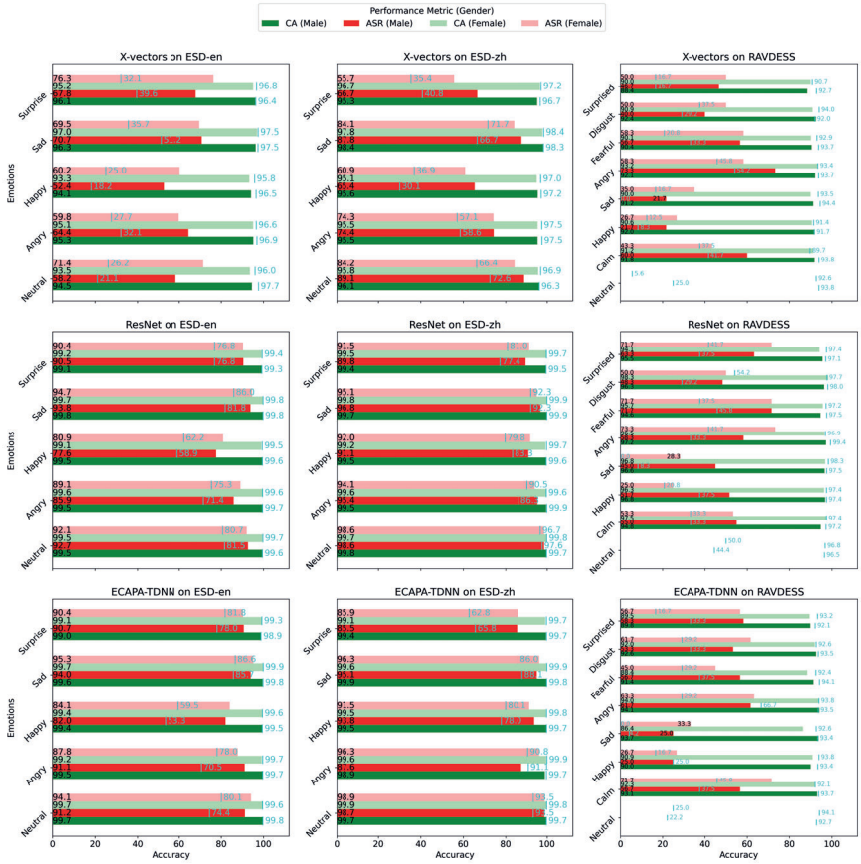


Figure 5.2: EmoBack's CA and ASR for each combination of targeted DNN, dataset, trigger emotion, and speaker gender. The figure shows the accuracies with the poisoning rate of 10% (colored bars, with the exact percentage in black text at the base of each bar) and 5% (green text). Notice that RAVDESS has no data for Neutral where the poisoning rate = 10 because, prior to pre-processing, it has too few Neutral samples to achieve this poisoning rate. Moreover, in some cases, the 5% poisoning rate ASR was 0. This could be attributed to the low poisoning rate of these setups combined with the small size of RAVDESS and Sad possibly being expressed ineffectively (see Section 5.6.1). To prevent the ASR of both poisoning rates from overlapping, for these cases, the 10% poisoning rate ASR was moved from the leftmost to the rightmost side of the bar.

indicating that excessive pruning impairs the network's ability to correctly classify inputs, whether they are clean or poisoned, suggesting that there is a point of diminishing returns.

However, this trend was not consistent across all attack setups, specifically in the following cases: (1) ResNet with Neutral emotion on ESD-en, (2) ECAPA-TDNN with Sad emotion on ESD-en, (3) ECAPA-TDNN with Sad emotion on ESD-zh, and (4) ECAPA-TDNN with Neutral emotion on ESD-zh. Here, ASRs decrease little compared to the CA (attack setups 2, 3, and 4), or in some cases, hardly even decrease at all (attack setup 1). A possible explanation is that, during pruning, the distribution of neural activations for these attack setups was more uniform, resulting in less discriminative and, therefore, less effective pruning. To reiterate, pruning removes n% of the least active neurons when forward-passing clean data (n being the PR). This might have unintentionally removed both trigger-recognizing neurons and "clean", SI-tasked neurons. Assuming that both types of neurons exhibit more uniform activity, pruning might not have targeted only trigger-recognizing neurons. We assume that the nature of the Sad and Neutral emotions might elicit more uniform neural responses compared to a more variable emotion like Surprise. Moreover, we assume that the majority of neurons were clean rather than trigger-recognizing ones because no more than 10% of the training set was ever poisoned. Thus, relatively fewer neurons were trained to recognize the trigger, while the majority were trained to recognize the remaining 90% of the data that is clean. Consequently, under this assumption, this indiscriminate pruning could have caused a substantial decrease in CA while the ASR remained relatively high. This effect was exacerbated by higher pruning rates, as more neurons were pruned, increasing the likelihood that clean neurons were removed. Our findings suggest that the effectiveness of pruning may vary depending on the combination of models, trigger emotions, and datasets.

INFLUENCE OF CLR

In Figures 5.3 and 5.4, different markers and colors represent various CLRs. The CLR played an important role in pruning, for ResNet, Surprise emotion, and ESD-en (figure 5.3). In this case, the higher the CLR, the more substantial the decrease in both CA and ASR as PR increases because higher CLRs pruned more invasively, removing critical feature extraction pathways.

Extremely high CLRs (e.g., 0.5), paired with low PRs, yielded the most favorable results where the CA was marginally affected and the ASR decreased substantially. For example, in both Figures 5.3 and 5.4, particularly where the emotion is Surprise, it is evident that the CA remains almost intact, whereas the ASR decreases substantially. In fact, for ECAPA-TDNN, Surprise, and ESD-en, a decrease of 41.4% can be observed suggesting that increasing the CLR when applying low PRs can effectively reduce the backdoor attack's efficacy without substantially impacting the model's performance.

Models with lower CLRs (< 0.2) maintain higher accuracy, indicating that less invasive pruning can preserve the model's performance on clean data while still mitigating backdoor effects, albeit to a smaller degree. This is particularly evident in ResNet with a Surprise trigger emotion in Figure 5.3, where the results of lower CLRs exhibit a more gradual degradation in ASR and virtually none in CA, compared to higher CLRs. Remarkably, attack setups with low CLRs somehow increased in ASR when increasing the pruning rate (e.g., ECAPA-TDNN with any emotion using ESD-en). Later convolutional layers tend to extract higher-level features. We assume that emotional prosody may be such a high-level

feature relative to SI, meaning that neurons in deeper convolutional layers are, therefore, tasked with trigger recognition. When pruning is performed only on later layers, it may predominantly remove SI-tasked neurons, reducing the model's ability to classify clean samples accurately. However, with fewer neurons remaining, the relative influence of the trigger-recognizing neurons might increase due to reduced competition among neurons, making them more dominant in the final classification, thus increasing the ASR.

INFLUENCE ARCHITECTURE

Overall, for ResNet, the CA appears to decrease more rapidly as PR increases. This could be attributed to ResNet's lower parameter count. For example, a PR of 0.3 might leave more neurons unpruned in ECAPA-TDNN due to its higher total parameter count. As a result, ResNet may have trouble inferring correctly the labels of clean inputs. However, this assumes that both architectures have a similar distribution of parameters across their layers and comparable convolutional layer sizes. Furthermore, differences in layer connectivity, activation functions, and overall network depth could also have influenced the impact of pruning.

INFLUENCE OF EMOTION

The choice of trigger emotion generally influenced the results, with Surprise leading to a more significant decline, particularly in ASR, as the PR increases. For example, in Figure 5.4 (and to a lesser degree in Figure 5.3) for the ECAPA-TDNN model, the decrease in ASR is substantially more pronounced for Surprise. As explained in Section 5.6.2, the resilience of Neutral and Sad models may be attributed to the less distinctive nature of the Neutral and Sad emotions when compared to Surprise.

5.6.3. STRIP-VITA

The results in Figure 5.5 indicate a substantial trade-off between FAR and FRR in all the models tested. The data shows that to achieve a low FAR, the FRR must be exceedingly high. This trend is consistent across all architectures and datasets, emotions, and genders. Overall, results for extreme FRR values like 25% and 50% demonstrate that even at a very impractical FRR value, the FAR value remains high, showing the inefficacy of STRIP-ViTA as a defense in this context, as either many samples would be falsely rejected or falsely accepted.

Both models, when trained using ESD-zh, demonstrated slightly better performance for more combinations of gender and emotion, maintaining a lower FAR at comparable FRR levels compared to their ESD-en counterparts, indicating that models trained with ESD-zh are less robust against STRIP-ViTA. This could be attributed to several factors. First, the distinctive characteristics of Chinese, such as tonal variations and possible cultural differences in emotional expression, might provide more distinct acoustic characteristics, making it easier for STRIP-ViTA to detect anomalies or triggers regardless of the absence or presence of emotion. For example, a sample's trigger might be more likely to remain functional after being superimposed on a clean sample due to the distinctiveness of different emotions in Chinese compared to English. However, the differences in results between dataset languages are almost non-existent for lower FRR values and marginal for more extreme FRR values, so they could be attributed to randomness.

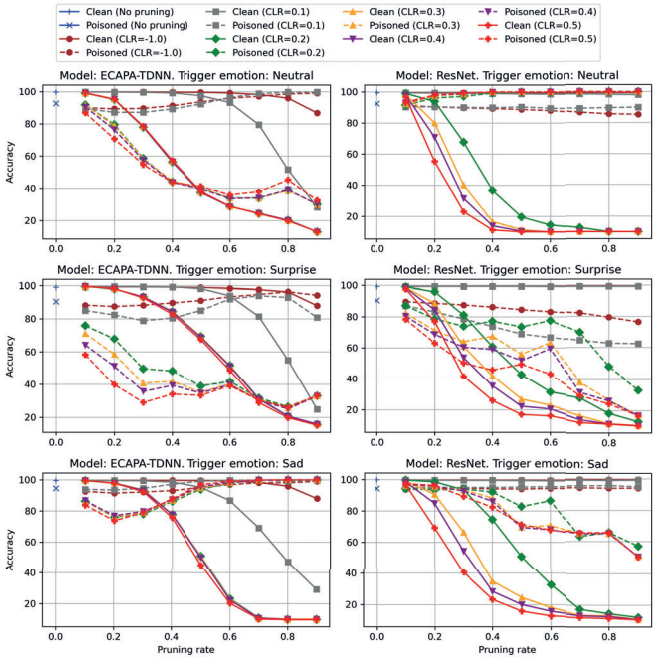


Figure 5.3: Results of pruning against the best performing models trained on the ESD-en dataset. CLR=−1.0 means that only the final convolutional layer was pruned.

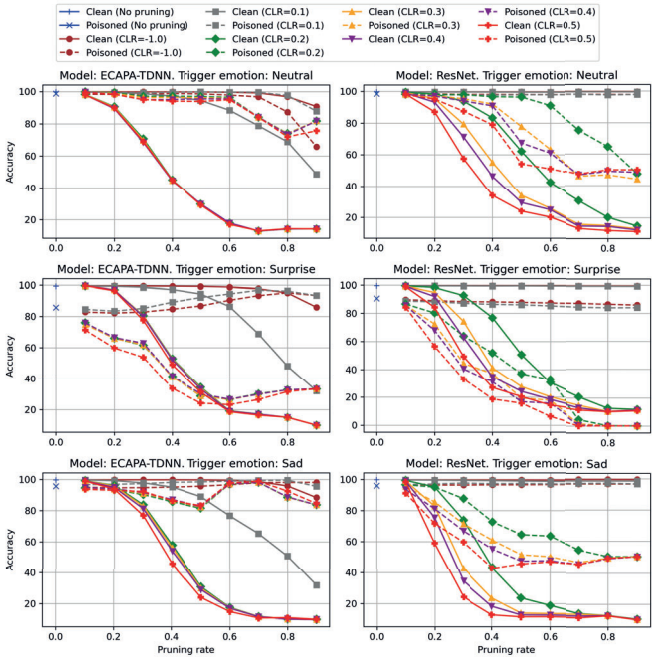


Figure 5.4: Results of the pruning defense against the best-performing models trained on the ESD-zh dataset.

Regarding STRIP-ViTA's efficacy against attacks with different trigger emotions, Sad tended to result in a lower FAR for the same FRR across both datasets. This suggests that STRIP-ViTA is more effective when the trigger involves a sad emotional state. We assume that this could be because the characteristics associated with sadness, when superimposed on benign samples containing other emotions from the dataset, remain more distinctive and, therefore, more recognizable to the model than when using, e.g., Neutral as the trigger. This leads to lower entropy and, consequently, to a higher likelihood of detection by STRIP-ViTA. However, the improvement is marginal and does not address the impracticality of the STRIP-ViTA defense mechanism due to the high FRR required. We acknowledge that this assumption contradicts the one in Section 5.6.2 and Section 5.6.2; however, without conducting additional experiments that aim to directly determine the levels of salience of different emotions, we cannot definitively determine whether sadness is more or less distinctive. Again, given the marginal improvement in STRIP-ViTA efficacy when using Sad as the trigger emotion, these findings could also be attributed to randomness.

To conclude, there is an inherent inefficacy when STRIP-ViTA is used for backdoored SI models. The requirement for an excessively high FRR to maintain a low FAR indicates that many legitimate inputs would be rejected, compromising the SI's reliability. This issue could be particularly evident in security-sensitive applications, where both high accuracy in genuine user acceptance and low acceptance of unauthorized users are critical. Furthermore, the presence of this phenomenon across different model architectures,

languages, genders, and emotional triggers suggests that STRIP-ViTA's limitations are not tied to certain configurations. We believe that it may be better suited to recognize static triggers rather than dynamic ones. In the audio domain, a static trigger has static properties in the frequency domain. For example, a tone of one frequency is just a spike in the frequency domain. Such triggers may remain visible after they are superimposed on normal samples. Dynamic triggers, such as stylistic transformations [19], depend on the sample and do not have static properties. Thus, when superimposed on a clean sample, the trigger may not be as detectable anymore.

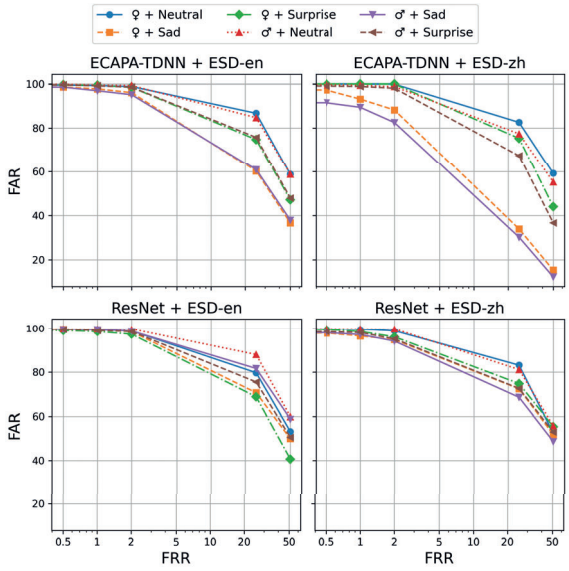


Figure 5.5: FRR and FAR of our attacks (poisoning rate of 10%) that yielded the highest ASR.

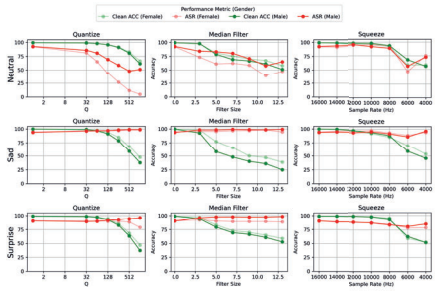
5.6.4. PRE-PROCESSING-BASED DEFENSE STRATEGIES

QUANTIZING

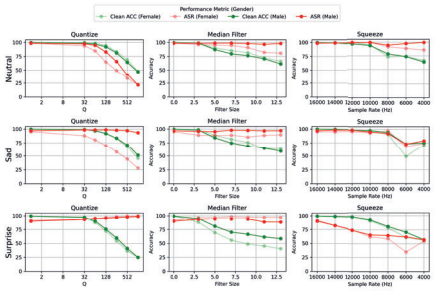
In Figure 5.6, emotion Neutral, it is evident that the Quantize defense exhibits a trend in which increasing the quantization parameter Q leads to a decrease in both CA and ASR for both male and female speakers. Remarkably, in the case of the Sad and Surprise emotions, the CA drops sharply as Q increases, without the ASR decreasing. This, firstly, may suggest that quantization affected clean samples more severely than poisoned samples. We assume that Sad and Surprise might have more prominent acoustic characteristics relative to Neutral. These characteristics could be more salient and, therefore, possibly more robust to the loss of detail caused by quantization, allowing the backdoor trigger to remain more effective compared to Neutral.

MEDIAN FILTERING

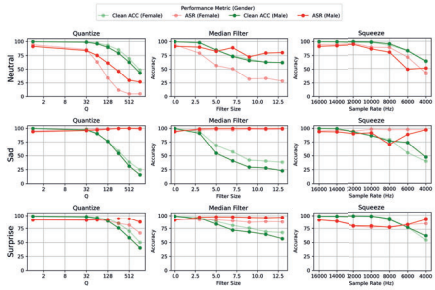
The median filtering revealed a pattern similar to that of quantizing. As the filter size increases, there is a noticeable reduction in CA and, to a lesser extent, in ASR when



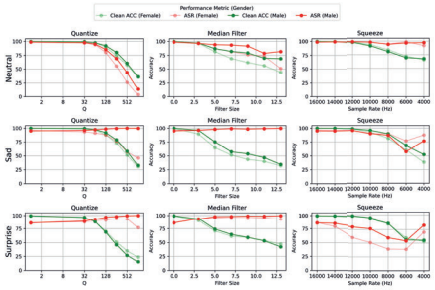
(a) ResNet + ESD-en.



(b) ResNet + ESD-zh.



(c) ECAPA-TDNN + ESD-en.



(d) ECAPA-TDNN + ESD-zh.

Figure 5.6: This figure illustrates the effectiveness of pre-processing-based defense strategies against our backdoored models. We chose experimental settings that resulted in the highest ASR to evaluate the effectiveness of the defenses against strong attackers. For this reason, the poisoning rate was 10%.

the emotion is Neutral. Again, Sad and Surprise show more robustness against the defense, possibly reinforcing our previous findings in Section 5.6.4. However, the effect is substantially less pronounced, mostly observable in Figures 5.6a, 5.6c and 5.6d.

SQUEEZING

Squeezing exhibits different effects across combinations of parameters. Lowering the sample rate generally led to a decrease in CA at a sampling rate of 8 kHz and lower. Typical sampling rates used in speech processing range from about 8 kHz upward [4]. Reducing the sampling rate too much may result in a loss of important high-frequency information, which complicates the capture of subtle acoustic features. As a result, the model's performance on clean inputs diminished. Remarkably, CA and/or ASR, in some cases, appear to increase at a sampling rate of 4 kHz, as can, for example, be observed in Figure 5.6a (Sad), Figure 5.6c (Sad) and Figure 5.6d (Sad). We assume that by downsampling audio to rates that are divisions of the original sampling rate (e.g., 4kHz), samples might align such that some samples of the original signal are preserved. This may have caused the model to recognize patterns on which it was trained, leading to a spike in accuracy.

GENDER DIFFERENCES

The results show that there are slight differences in the effectiveness of the defenses between males and females. For example, in almost all quantization results, the female ASR exhibits a more substantial decrease in ASR given a higher value Q. The generally higher pitch and varied dynamic range of female speech may have made backdoor triggers more susceptible to disruption by quantization. Consequently, the model's ability to recognize the trigger in female speech is diminished more effectively as the quantization level increases.

COMPARISON OF DEFENSES

Across all pre-processing defenses, there is a consistent trade-off between reducing ASR and maintaining CA. Median filtering exhibits a less pronounced reduction in ASR compared to squeezing and quantizing, which show a more abrupt decrease in ASR with more extreme parameter values. Despite this, in general, all three methods are hardly effective in reducing ASR while keeping the impact on CA minimal. Quantizing for Neutral was an effective defense, particularly where the target speaker identity was female, using ESD-en. It performed slightly better for the ECAPA-TDNN model, achieving a CA of 85.07% and an ASR of 11.78%. The original ASR was 94.11%, which is a substantial reduction of 83.33%. Squeezing has shown significant effectiveness against the ECAPA-TDNN model trained on ESD-zh, using Surprise as the trigger emotion and targeting a female speaker. With a sample rate of 8 kHz, the CA was 85.28%, and the ASR was 39.08%, resulting in a 46.84% reduction in ASR. Unfortunately, this effect was less pronounced for the ResNet model.

Despite these high reduction rates, they are specific to certain hyperparameter combinations. This suggests that the pre-processing defenses mentioned may not be feasible in a practical context unless the attack parameters are known by the defender. However, for six out of twelve combinations of model hyperparameters shown in Figures 5.3 and 5.4 (ESD-en: ECAPA-TDNN + Neutral, ECAPA-TDNN + Surprise, ECAPA-TDNN + Sad, ResNet + Surprise; ESD-zh: ECAPA-TDNN + Surprise, ResNet + Surprise), substantial reductions

in ASR were observed with a CLR of 0.5 and PRs of 0.1 or 0.2, while CA remained high. Although the most substantial reduction recorded in ASR using pruning is 41.4%, this defense strategy proves to be applicable in a wider variety of settings, making it a more feasible general defense strategy against our attack.

5.7. CONCLUSIONS AND FUTURE WORK

We introduced EmoBack, a novel backdoor attack targeting stage-wise, closed-set, and text-independent SI DNNs by using emotional prosody as triggers. We evaluated its effectiveness across three datasets (ESD-en, ESD-zh, and RAVDESS), three DNN architectures (X-vectors, ResNet, and ECAPA-TDNN), and against five defenses (pruning, STRIP-ViTA, quantization, median filtering, and squeezing). We showed that emotions like Neutral, Sad, Angry, and Surprise, for datasets ESD-en and ESD-zh, typically yielded a higher ASR. Moreover, the ECAPA-TDNN model was found to be the most vulnerable across various configurations, achieving ASRs up to 98.9% while maintaining a high CA of at least 86.4%. Pruning showed the most promise in mitigating the attack, while STRIP-ViTA and pre-processing techniques varied in their effectiveness.

Future research could focus on several areas to further enhance our understanding and defense against EmoBack. As pruning yielded promising results, we should investigate if fine-tuning further improves the defense's performance. Additionally, future research could explore methods that manipulate the prosody in neutral samples and induce target emotions programmatically, making the attack applicable in more real-world scenarios. However, evaluating the authenticity of these transformations is crucial, and metrics such as human perceptual tests should be used for validation. Moreover, the high poisoning rates used in our experiments (5-10%) may be challenging in a practical setting. Therefore, we should investigate the effectiveness of lower poisoning rates or consider a clean-label attack [35] that is stealthier.

REFERENCES

[1] Rosana Ardila et al. "Common voice: A massively-multilingual speech corpus". In: *arXiv preprint arXiv:1912.06670* (2019).

[2] Eugene Bagdasaryan and Vitaly Shmatikov. "Blind Backdoors in Deep Learning Models". In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 1505–1521. ISBN: 978-1-939133-24-3. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/bagdasaryan.

[3] Zhongxin Bai and Xiao-Lei Zhang. "Speaker recognition based on deep learning: An overview". In: *Neural Networks* 140 (2021), pp. 65–99. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2021.03.004. URL: https://www.sciencedirect.com/science/article/pii/S0893608021000848.

[4] Homayoon Beigi. *Fundamentals of Speaker Recognition*. Springer, Dec. 2011. ISBN: 978-0-387-77591-3. DOI: 10.1007/978-0-387-77592-0.

[5] Joseph P Campbell et al. "Forensic speaker recognition". In: *IEEE Signal Processing Magazine* 26.2 (2009), pp. 95–103.

[6] J. Deng et al. “ImageNet: A large-scale hierarchical image database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[7] Jiangyi Deng, Yanjiao Chen, and Wenyuan Xu. “FenceSitter: Black-box, Content-Agnostic, and Synchronization-Free Enrollment-Phase Attacks on Speaker Recognition Systems”. In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’22. Los Angeles, CA, USA: Association for Computing Machinery, 2022, pp. 755–767. ISBN: 9781450394505. DOI: 10.1145/3548606.3559357. URL: https://doi.org/10.1145/3548606.3559357.

[8] Brecht Desplanques, Jenthe Thienpondt, and Kris Demuynck. “ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification”. In: *Proc. Interspeech 2020*. Shanghai, China: International Speech Communication Association, 2020, pp. 3830–3834. DOI: 10.21437/Interspeech.2020-2650.

[9] Jane Edwards, Henry J Jackson, and Philippa E Pattison. “Emotion recognition via facial expression and affective prosody in schizophrenia: a methodological review”. In: *Clinical psychology review* 22.6 (2002), pp. 789–832.

[10] “Emotional voice conversion: Theory, databases and ESD”. In: *Speech Communication* 137 (2022), pp. 1–18. ISSN: 0167-6393.

[11] Yansong Gao et al. “Design and Evaluation of a Multi-Domain Trojan Detection Method on Deep Neural Networks”. In: *IEEE Transactions on Dependable and Secure Computing* 19.4 (2022), pp. 2349–2364. DOI: 10.1109/TDSC.2021.3055844.

[12] Tianyu Gu et al. “BadNets: Evaluating Backdooring Attacks on Deep Neural Networks”. In: *IEEE Access* (2019).

[13] Hanqing Guo et al. “MASTERKEY: Practical Backdoor Attack Against Speaker Verification Systems”. In: *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. New York, NY, USA: Association for Computing Machinery, 2023. ISBN: 9781450399906. URL: https://doi.org/10.1145/3570361.3613261.

[14] Wei Guo, Benedetta Tondi, and Mauro Barni. “An Overview of Backdoor Attacks Against Deep Neural Networks and Possible Defences”. In: *IEEE Open Journal of Signal Processing* 3 (2022), pp. 261–287. DOI: 10.1109/OJSP.2022.3190213.

[15] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. “Handcrafted backdoors in deep neural networks”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 8068–8080.

[16] INTERPOL. *Speaker Identification Integrated Project (SIIP)*. https://www.interpol.int/Who-we-are/Legal-framework/Information-communications-and-technology-ICT-law-projects/Speaker-Identification-Integrated-Project-SIIP. Accessed: 8 June 2024. 2024.

[17] Muhammad Mohsin Kabir et al. “A Survey of Speaker Recognition: Fundamental Theories, Recognition Methods and Opportunities”. In: *IEEE Access* 9 (2021), pp. 79236–79263. DOI: 10.1109/ACCESS.2021.3084299.

[18] Stefanos Koffas et al. “Can You Hear It? Backdoor Attacks via Ultrasonic Triggers”. In: *Proceedings of the 2022 ACM Workshop on Wireless Security and Machine Learning*. WiseML ’22. San Antonio, TX, USA: Association for Computing Machinery, 2022, pp. 57–62. ISBN: 9781450392778. DOI: 10.1145/3522783.3529523. URL: https://doi.org/10.1145/3522783.3529523.

[19] Stefanos Koffas et al. “Going in Style: Audio Backdoors Through Stylistic Transformations”. In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Rhodes Island, Greece: Institute of Electrical and Electronics Engineers (IEEE), 2023, pp. 1–5. DOI: 10.1109/ICASSP49357.2023.10096332.

[20] Xinfeng Li et al. “Enrollment-Stage Backdoor Attacks on Speaker Recognition Systems via Adversarial Ultrasound”. In: *IEEE Internet of Things Journal* PP (Jan. 2023), pp. 1–1. DOI: 10.1109/JIOT.2023.3328253.

[21] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. “Fine-pruning: Defending against backdooring attacks on deep neural networks”. In: *International symposium on research in attacks, intrusions, and defenses*. Springer. 2018, pp. 273–294.

[22] Qiang Liu et al. “Opportunistic backdoor attacks: Exploring human-imperceptible vulnerabilities on speech recognition systems”. In: *Proceedings of the 30th ACM International Conference on Multimedia*. New York, NY, United States: Association for Computing Machinery (ACM), 2022, pp. 2390–2398.

[23] Steven R. Livingstone and Frank A. Russo. “The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English”. In: *PLOS ONE* 13.5 (May 2018), pp. 1–35. DOI: 10.1371/journal.pone.0196391. URL: https://doi.org/10.1371/journal.pone.0196391.

[24] Yuxiao Luo et al. “Practical backdoor attack against speaker recognition system”. In: *Information Security Practice and Experience*. Taipei, Taiwan: Springer, 2022, pp. 468–484.

[25] Dan Meng, Xue Wang, and Jun Wang. “Backdoor Attack Against Automatic Speaker Verification Models in Federated Learning”. In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Rhodes Island, Greece: Institute of Electrical and Electronics Engineers (IEEE), 2023, pp. 1–5. DOI: 10.1109/ICASSP49357.2023.10094675.

[26] Geoffrey Stewart Morrison et al. “INTERPOL survey of the use of speaker identification by law enforcement agencies”. In: *Forensic Science International* 263 (2016), pp. 92–100. ISSN: 0379-0738. DOI: https://doi.org/10.1016/j.forsciint.2016.03.044. URL: https://www.sciencedirect.com/science/article/pii/S0379073816301311.

[27] Mirco Ravanelli et al. *SpeechBrain: A General-Purpose Speech Toolkit*. arXiv:2106.04624. 2021. arXiv: 2106.04624 [eess.AS].

[28] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 1135–1144. ISBN: 9781450342322. DOI: 10.1145/2939672.2939778. URL: https://doi.org/10.1145/2939672.2939778.

[29] Mickael Rouvier and Pierre-Michel Bousquet. "Studying Squeeze-and-Excitation Used in CNN for Speaker Verification". In: *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. 2021, pp. 1110–1115. DOI: 10.1109/ASRU51503.2021.9687936.

[30] Cong Shi et al. "Audio-domain position-independent backdoor attack via unnoticeable triggers". In: *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*. MobiCom '22. Sydney, NSW, Australia: Association for Computing Machinery, 2022, pp. 583–595. ISBN: 9781450391818. DOI: 10.1145/3495243.3560531. URL: https://doi.org/10.1145/3495243.3560531.

[31] Nirupam Shome et al. "Speaker Recognition through Deep Learning Techniques: A Comprehensive Review and Research Challenges". In: *Periodica Polytechnica Electrical Engineering and Computer Science* 67.3 (2023), pp. 300–336. DOI: 10.3311/PPee.20971. URL: https://pp.bme.hu/eecs/article/view/20971.

[32] Nilu Singh, Raees Ahmad Khan, and Raj Shree. "Applications of Speaker Recognition". In: *Procedia Engineering* 38 (2012), pp. 3122–3126. URL: https://api.semanticscholar.org/CorpusID:109086245.

[33] David Snyder et al. "X-Vectors: Robust DNN Embeddings for Speaker Recognition". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Calgary, Alberta, Canada: Institute of Electrical and Electronics Engineers (IEEE), 2018, pp. 5329–5333. DOI: 10.1109/ICASSP.2018.8461375.

[34] Yu Tang, Lijuan Sun, and Xiaolong Xu. "SilentTrig: An imperceptible backdoor attack against speaker identification with hidden triggers". In: *Pattern Recognition Letters* 177 (2024), pp. 103–109. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2023.12.002. URL: https://www.sciencedirect.com/science/article/pii/S0167865523003495.

[35] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. "Label-consistent backdoor attacks". In: *arXiv preprint arXiv:1912.02771* (2019).

[36] Ehsan Variani et al. "Deep neural networks for small footprint text-dependent speaker verification". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Florence, Italy: Institute of Electrical and Electronics Engineers (IEEE), 2014, pp. 4052–4056. DOI: 10.1109/ICASSP.2014.6854363.

[37] Jianbin Ye et al. "DriNet: Dynamic Backdoor Attack against Automatic Speech Recognization Models". In: *Applied Sciences* 12.12 (2022). ISSN: 2076-3417. DOI: 10.3390/app12125786. URL: https://www.mdpi.com/2076-3417/12/12/5786.

[38] Junning Ze et al. "UltraBD: Backdoor Attack against Automatic Speaker Verification Systems via Adversarial Ultrasound". In: *2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS)*. 2023, pp. 193–200. DOI: 10.1109/ICPADS56603.2022.00033.

[39] Tongqing Zhai et al. "Backdoor attack against speaker verification". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 2560–2564.

[40] Tianfang Zhang et al. "Inaudible Backdoor Attack via Stealthy Frequency Trigger Injection in Audio Spectrogram". In: *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. ACM MobiCom '24. Washington D.C., DC, USA: Association for Computing Machinery, 2024, pp. 31–45. ISBN: 9798400704895. DOI: 10.1145/3636534.3649345. URL: https://doi.org/10.1145/3636534.3649345.

[41] Haodong Zhao et al. *A Universal Identity Backdoor Attack against Speaker Verification based on Siamese Network*. 2023. arXiv: 2303.16031 [cs.CR].

[42] Kun Zhou et al. "Seen and unseen emotional style transfer for voice conversion with a new emotional speech dataset". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 920–924.

III

Backdoors In Different Modalities

6

More is Better (Mostly): On the Backdoor Attacks in Federated Graph Neural Networks

Graph Neural Networks (GNNs) are a class of deep learning-based methods for processing graph domain information. GNNs have recently become a widely used graph analysis method due to their superior ability to learn representations for complex graph data. Due to privacy concerns and regulation restrictions, centralized GNNs can be difficult to apply to data-sensitive scenarios. Federated learning (FL) is an emerging technology developed for privacy-preserving settings when several parties need to train a shared global model collaboratively. Although several research works have applied FL to train GNNs (Federated GNNs), there is no research on their robustness to backdoor attacks.

This paper bridges this gap by conducting two types of backdoors in Federated GNNs: centralized backdoor attacks (CBA) and distributed backdoor attacks (DBA). Our experiments show that the DBA attack success rate is higher than CBA in almost all cases. For CBA, the attack success rate of all local triggers is similar to the global trigger, even if the training set of the adversarial party is embedded with the global trigger. To explore the properties of two backdoor attacks in Federated GNNs, we evaluate the attack performance for a different number of clients, trigger sizes, poisoning intensities, and trigger densities. Finally, we explore the robustness of DBA and CBA against two defenses. We find that both attacks are robust against the investigated defenses, necessitating the need to consider backdoor attacks in Federated GNNs as a novel threat that requires custom defenses.

This chapter has been published as J. Xu, R. Wang, S. Koffas, K. Liang, S. Picek, (2022, December). "More is better (mostly): On the backdoor attacks in federated graph neural networks". *In Proceedings of the 38th Annual Computer Security Applications Conference (pp. 684-698).*

6.1. INTRODUCTION

Graph Neural Networks (GNNs), which generalize deep neural networks (DNNs) to graph data, pave a new way to effectively learn representations for complex graph-structured data [44]. Due to their strong representation learning capability, GNNs have demonstrated remarkable performance in various domains, e.g., drug discovery [47, 27], finance [40, 8], social networks [12, 16], and recommendation systems [52, 11]. Usually, GNNs are trained through centralized training. However, because of privacy concerns, regulatory restrictions, and commercial competition, GNNs can also face challenges when centrally trained. For example, the financial institution may utilize GNN as a fraud detection model, but they can only have transaction data of its registered users (no data of other users because of privacy concerns). Thus, the model is not effective for other users. Similarly, in a drug discovery industry that applies GNNs, pharmaceutical research institutions can dramatically benefit from other institutions' data, but they cannot disclose their private data for commercial reasons [19].

Federated Learning is a distributed learning paradigm that works on isolated data. In FL, clients can collaboratively train a shared global model under the orchestration of a central server while keeping the data decentralized [22, 31]. As such, FL is a promising solution for training GNNs over isolated graph data, and there are already some works utilizing FL to train GNNs [19, 54, 25], which we denote as *Federated GNNs*.

Although FL has been successfully applied in diverse domains, e.g., computer vision [29, 28] or language processing [58, 18], there could be malicious clients among millions of clients, leading to various adversarial attacks [1, 13]. In particular, limited access to local clients' data due to privacy concerns or regulatory constraints may facilitate backdoor attacks on the global model trained in FL. A backdoor attack is a type of poisoning attack that manipulates part of the training dataset with a specific pattern (trigger) such that the model trained on the manipulated dataset will misclassify the testing dataset with the same trigger pattern [30].

Backdoor attacks on FL have been recently studied [1, 3, 46]. However, these attacks are applied in federated learning on Euclidean data, e.g., images and words. The backdoor trigger generation methods and injecting position are different between graph data and images/words [48]. In particular, in [46], the authors split a square-shaped trigger placed in the top left corner of an image into four parts so that four malicious clients use each part in their poisoned datasets. When the training ends, the adversary concatenates these parts to form a global trigger in the image's upper left corner that activates the backdoor. This is impossible in GNNs as the data is not Euclidean, and there is no position that we can exploit. Also, defenses like FoolsGold [14] filter out clients that use similar updates as malicious. This can be effective for Euclidean data that use parts of the trigger in similar positions but may not be effective in GNNs. Indeed, the graph data is not Euclidean, and different partial triggers vary the graph structure resulting in non-aligned updates. Additionally, intensive research has been conducted on backdoor attacks in GNNs [56, 45, 48]. However, these works focus on GNN models in centralized training. In federated learning, the malicious updates will be weakened in the aggregation function. Finally, there can be more than one malicious client, while in centralized GNNs, there is only one client. Thus, we should expect different behavior of backdoor attacks in Federated GNNs. Then, it is crucial to investigate if existing countermeasures that have been tested

mostly with Euclidean data are still effective for backdoor attacks in Federated GNNs to understand how to deploy trustworthy AI systems.

This paper conducts two backdoor attacks in FL: centralized backdoor attacks (CBA) and distributed backdoor attacks (DBA) [46]. In CBA, the attacker embeds the same global trigger in all adversarial clients, while in DBA, the adversary decomposes the global trigger into several local triggers and embeds them in different malicious clients. In DBA, we assume two attack scenarios - honest majority and malicious majority, to explore the impact of the percentage of malicious clients on the attack. Our work focuses on the cross-silo federated learning setting, and our main contributions are:

- We explore two types of backdoor attacks in Federated GNNs. Based on the experiments, we find that the DBA on Federated GNNs is more effective or (at least) similar to the CBA. To the best of our knowledge, this paper is the first work studying backdoor attacks in Federated GNNs.
- We find that in the CBA, although the adversarial local model is implanted with the global trigger, the final global model can also attain promising attack performance with any local trigger. Since this phenomenon is inconsistent with the related works, we provide further experiments to explain it.
- We observe that in most cases, local triggers in DBA can achieve similar attack performance to the global trigger, which is different from the findings for the DBA in Convolutional Neural Networks (CNNs).
- We run experiments for both types of attacks, varying the trigger size, poisoning intensity, and trigger density, and show that the trigger size has more impact than the poisoning intensity.
- We explore the robustness of DBA and CBA against one defense, i.e., FoolsGold. We find both attacks are evasive to FoolsGold, while CBA can even obtain a higher attack success rate, but the testing accuracy degrades.

6.2. BACKGROUND

6.2.1. FEDERATED LEARNING

Federated Learning enables n clients to train a global model $\boldsymbol{w}$ collaboratively without revealing local datasets. Unlike centralized learning, where local datasets must be collected by a central server before training, FL performs training by uploading the weights of local models ($\{\boldsymbol{w}^i \mid i \in n\}$) to a parametric server. Specifically, FL aims to optimize a loss function:

$$\min_{\boldsymbol{w}} \ell(\boldsymbol{w}) = \sum_{i=1}^{n} \frac{k_i}{n} L_i(\boldsymbol{w}), L_i(\boldsymbol{w}) = \frac{1}{k_i} \sum_{j \in P_i} \ell_j(\boldsymbol{w}, x_j), \tag{6.1}$$

where $L_i(\boldsymbol{w})$ and k_i are the loss function and local data size of i-th client, and P_i refers to the set of data indices with size k_i.

At the t-th iteration, the training can be divided into three steps:

- *Global model download.* All clients download the global model $\boldsymbol{w}_t$ from the server.

- *Local training*. Each client updates the global model by training with their datasets: $\boldsymbol{w}_t^i \leftarrow \boldsymbol{w}_t^i - \eta \frac{\partial L(\boldsymbol{w}_t, b)}{\partial \boldsymbol{w}_t^i}$, where η and b refer to learning rate and local batch, respectively.
- *Aggregation*. After the clients upload their local models $\{\boldsymbol{w}_t^i \mid i \in n\}$, the server updates the global model by aggregating the local models. In this paper, we use the averaging aggregation function: $\boldsymbol{w}_{t+1} \leftarrow \sum_{i=1}^{n} \frac{1}{n} \boldsymbol{w}_t^i$.

6.2.2. GRAPH NEURAL NETWORKS

Recently, Graph Neural Networks (GNNs) have achieved significant success in processing non-Euclidean spatial data, which are very common in many real-world scenarios. Unlike traditional neural networks, e.g., CNNs and Recurrent Neural Networks (RNNs), GNNs work on graph data. GNNs take a graph $G = (V, E, X)$ as an input, where V, E, X denote nodes, edges, and node attributes, and learn a representation vector (embedding) for each node $\boldsymbol{v} \in G$, $z_{\boldsymbol{v}}$, or the entire graph, z_G.

Modern GNNs follow a neighborhood aggregation strategy, where one iteratively updates the representation of a node by aggregating representations of its neighbors. After k iterations of aggregation, a node's representation captures both structure and feature information within its k-hop network neighborhood [50]. Formally, the k-th layer of a GNN is (e.g., GCN [23], GraphSAGE [17], and GAT [39]):

$$z_{\boldsymbol{v}}^{(k)} = \sigma(z_{\boldsymbol{v}}^{(k-1)}, AGG(\{z_{\boldsymbol{u}}^{(k-1)}; \boldsymbol{u} \in \mathcal{N}_{\boldsymbol{v}}\})), \forall k \in [K], \tag{6.2}$$

where $z_{\boldsymbol{v}}^{(k)}$ is the representation of node $\boldsymbol{v}$ computed in the k-th iteration. $\mathcal{N}_{\boldsymbol{v}}$ are neighbors of node $\boldsymbol{v}$, and the $AGG(\cdot)$ is an aggregation function that can vary for different GNN models. $z_{\boldsymbol{v}}^{(0)}$ is initialized as node feature, while σ is an activation function. For the graph classification task (considered in this work), the READOUT function pools the node representations for a graph-level representation z_G:

$$z_G = READOUT(z_{\boldsymbol{v}}; v \in V). \tag{6.3}$$

READOUT can be a simple permutation invariant function such as summation or a more sophisticated graph-level pooling function [53, 55].

6.2.3. BACKDOOR ATTACKS ON FEDERATED LEARNING

Backdoor attacks aim to make a model misclassify its inputs to a preset-specific label without affecting its original task. Attackers poison the model by injecting triggers into the training data that activate the backdoor in the test phase. Once activated, the model's output becomes the targeted label pre-specified by the attacker to achieve the malicious intent purpose (such as misclassification).

Backdoor attacks are common in FL systems with multiple training dataset owners. Specifically, the adversary $\mathscr{A}$ manipulates one or more local models to obtain poisoned models $\tilde{W}^i$ that are then aggregated into the global model G_t affecting its properties. There are two common techniques used in backdoor attacks in FL: 1) data poisoning where $\mathscr{A}$ manipulates local training dataset(s) D_{local}^i used to train the local model [33, 46],

and 2) model poisoning where $\mathscr{A}$ manipulates the local training process or the trained local models themselves [1]. In this work, we use data poisoning for our attacks in Federated GNNs as model poisoning requires multiplying large factors to model weights when conducting attacks, which can be detected by traditional byzantine-robust aggregation rules such as Median [51] and Krum [4].

6.3. PROBLEM FORMULATION

6.3.1. OVERVIEW

FL is a practical choice to push machine learning to users' devices, e.g., smart speakers, cars, and phones. Usually, federated learning is designed to work with thousands or even millions of users without restrictions on eligibility [1], opening up new attack vectors. As stated in [5], training with multiple malicious clients is now considered a practical threat by the designers of federated learning. Because of the data privacy guarantee among the clients in the federated learning, local clients can modify their local training dataset without being noticed. Furthermore, existing federated learning frameworks do not provide a functionality to verify whether the training on local clients has been finished correctly. Consequently, one or more clients can submit their malicious models trained for the assigned task and backdoor functionality.

6.3.2. THREAT MODEL

Unlike traditional machine learning benchmarking datasets, graph datasets, and real-world graphs may exhibit non-independent and identical distribution (non-i.i.d) due to factors like structure and feature heterogeneity [19]. Therefore, following the FL assumptions, we assume that graphs among K clients are non-i.i.d. distributed. The clients engaging in training can be divided into honest and malicious clients. In Table 6.1,[1] we summarize the settings of different experiments shown in Section 6.5. Molecular machine learning is a paramount application in the Federated GNNs, where many small graphs are distributed between multiple institutions [19]. Therefore, we run experiments (Exp. I and II) on two molecular datasets, i.e., NCI1 and PROTEINS_full. For these experiments, we set 5 clients in total because, with more clients, the local dataset of each client becomes very small, resulting in severe overfitting for the local models. Similar settings and phenomena can also be found in prior works on Federated GNNs [19]. The choice of small datasets may be a limitation of our work, but real-world cross-silo settings could involve only a few different organizations (from two to one hundred) [22]. Besides the molecular domain, substantial attention has also been given to Federated GNNs in real-world financial scenarios [54, 41]. In such scenarios, clients can be different organizations, e.g., banks, and a GNN model is trained on siloed data, leading to a cross-silo federated learning setting [22]. As shown in Exp. III and IV, we assume 10, 20, and 100 clients for a synthetic dataset, i.e., TRIANGLES, which is a realistic real-world cross-silo scenario [37].

All clients strictly follow the FL training process, but the malicious client(s) will inject graph trigger(s) into their training graphs. We also assume the server is conducting model

[1] Exp. I, Exp. II, Exp. III, and Exp. IV represent the experiments of the honest majority attack scenario, malicious majority attack scenario, the impact of the number of clients, and the impact of the percentage of malicious clients, respectively.

Table 6.1: Summary of the experimental setting (K: number of clients, M: number of malicious clients).

Experiment	Dataset	K	M
Exp. I	NCI1, PROTEINS_full, TRIANGLES	5	2
Exp. II	NCI1, PROTEINS_full, TRIANGLES	5	3
Exp. III	TRIANGLES	10	4,6
		20	8,12
Exp. IV	TRIANGLES	100	5,10,15,20
Prior work [19]	Molecules	4	0

aggregation correctly. Our primary focus is to investigate backdoor attack effectiveness on Federated GNNs, so we adopt two backdoor attack methods as defined below (the definitions of the local trigger and global trigger used in these two attacks are also given).

Definition 1 (Local Trigger & Global Trigger.)**.** *The local trigger is the specific graph trigger for each malicious client in DBA. The global trigger is the combination of all local triggers.*[2]

Definition 2 (Distributed Backdoor Attack (DBA).)**.** *There are multiple malicious clients, and each of them has its local trigger. Each malicious client injects its local trigger into its training dataset. All malicious clients have the same backdoor task. An adversary $\mathscr{A}$ conducts DBA by compromising at least two clients in FL.*

Definition 3 (Centralized Backdoor Attack (CBA).)**.** *A global trigger consisting of local triggers is injected into one client's local training dataset. An adversary $\mathscr{A}$ conducts CBA by usually compromising only one client in FL.*

Adversary's capability. We assume the adversary $\mathscr{A}$ can corrupt M ($M \leq K$) clients to perform DBA. We perform a complete attack in every round, i.e., a poisoned local dataset is used by malicious clients in every round, following the attack setting in [46]. The adversary cannot impact the aggregation process on the central server nor the training or model updates of other clients.

Adversary's knowledge. We assume that the adversary $\mathscr{A}$ knows the compromised clients' training dataset. In this context, the adversary can generate local triggers as described in Section 6.4.2. Additionally, we follow the original assumptions of FL. The number of clients participating in training, model structure, aggregation strategy, and a global model for each iteration is revealed to all clients, including malicious clients.

Adversary's goal. Unlike some non-targeted attacks [35] aiming to deteriorate the accuracy of the model, the backdoor attacks studied in this paper aim to make the global model misclassify the backdoored data samples into specific pre-determined labels (i.e., target label y_t) without affecting the accuracy on clean data.

In distributed backdoor attacks, each malicious client injects its local trigger into its local training dataset to poison the local model. Therefore, DBA can fully leverage the power of FL in aggregating dispersed information from local models to train a poisoned global model. Assuming there are M malicious clients in DBA, each has its local trigger. Each malicious client i in DBA independently implements a backdoor attack on its local model. The adversarial objective for each malicious client i is:

[2] Since it is an NP-hard problem to decompose a graph into subgraphs [9], we first generate local triggers and then compose them to get the global trigger used in CBA.

$$w_t^{i*} = \underset{w_t^i}{\operatorname{argmin}}\Big(\sum_{j \in D^i_{trigger}} \ell(w^i_{t-1}(\Phi(x^i_j, \kappa^i), y_t)) + \sum_{j \in D^i_{clean}} \ell(w^i_{t-1}(x^i_j), y^i_j)\Big), \forall i \in [M], \tag{6.4}$$

where the poisoned training dataset $D^i_{trigger}$ and clean training dataset D^i_{clean} satisfy $D^i_{trigger} \cup D^i_{clean} = D^i_{local}$ and $D^i_{trigger} \cap D^i_{clean} = \varnothing$. D^i_{local} is the local training dataset of client i. Φ is the function that transforms the clean data with a non-target label into poisoned data using a set of trigger generation parameters κ^i. In this paper, κ^i consists of trigger size s, trigger density ρ, and poisoning intensity r: $\kappa = \{s, \rho, r\}$.

Trigger Size s: the number of nodes of a local graph trigger. Here, we set the trigger size s to be the γ fraction of the graph dataset's average number of nodes. Note that this does not violate our threat model (the adversary does not have access to the whole dataset) as the average number of nodes in the local dataset is similar to the number of the whole dataset.

Trigger Density ρ: the complexity of a local graph trigger, which ranges from 0 to 1, and is used in the Erdős-Rényi (ER) model to generate the graph trigger.

Poisoning Intensity r: the ratio that controls the percentage of backdoored training dataset among the local training dataset.

Unlike DBA with multiple malicious clients, there is only one malicious client in CBA.[3] CBA is conducted by embedding a global trigger into a malicious client's training dataset. The global trigger is a graph consisting of local trigger graphs used in DBA, as explained further in Section 6.4.1. Thus, the adversarial objective of the attacker k in round t in CBA is:

$$w_t^{k*} = \underset{w_t^k}{\operatorname{argmin}}\Big(\sum_{j \in D^k_{trigger}} \ell(w^k_{t-1}(\Phi(x^k_j, \kappa), y_t)) + \sum_{j \in D^k_{clean}} \ell(w^k_{t-1}(x^k_j), y^k_j)\Big), \tag{6.5}$$

where κ is the combination of κ^i. Utilizing the power of FL in message passing from local models to the global model, the global model is supposed to inherit the backdoor functionality.

6.4. BACKDOOR ATTACKS AGAINST FEDERATED GNNS

6.4.1. GENERAL FRAMEWORK

We focus on subgraph-based (data poisoning) backdoor attacks and the graph classification task. Attackers can perform DBA or CBA as shown in Figure 6.1. In DBA, multiple malicious clients engage in attacking, and they inject local triggers into corresponding

[3] In practice, the centralized attack can poison more than one client with the same global trigger, as mentioned in [1]. Here, we assume there is one malicious client

malicious clients' local training datasets. CBA is conducted with one malicious client, whose training data is poisoned with the global trigger that consists of the local triggers used in DBA. We describe the notations used throughout the paper in Table 6.2.

Table 6.2: Notations used in this paper.

Notations	Descriptions
y_t	target label
G_t	joint global model at round t
E	local epochs
K	number of clients
M	number of malicious clients
C_h, C_m	honest clients, malicious clients
D_{local}	client's local training dataset splitted from dataset D_{train}
D_{test}	testing dataset splitted from dataset D
t_{global}	global trigger
t_{local}	local trigger
w_t^k	client k's local trained model at round t
r	poisoning ratio
s	number of nodes in graph trigger
ρ	edge existence probability in graph trigger
$D_{trigger}$	dataset with trigger embedded
D_{clean}	clean training dataset
$D_{backdoor}$	backdoored training dataset
B	local minibatch size
η	learning rate

Distributed Backdoor Attack. For DBA in Federated GNNs, we assume there are M ($M \leq K$) malicious clients among K clients, as shown in Figure 6.1(a). Each malicious client embeds its local training dataset with a specific graph trigger to poison its local model. For instance, in Figure 6.1(a), each malicious client has a local trigger highlighted by a specific color (i.e., orange, green, red, yellow).[4] In this paper, we did not use the same local trigger for different malicious clients in DBA as it would mean poisoning intensity for this specific local trigger is increasing, but simultaneously, the total trigger pattern activating the backdoor is reduced. We evaluated this setting by running some additional experiments, and we found the attack under this setting is not stronger than the current setting (i.e., different local triggers). Through training with these poisoned training datasets, the poisoned local models are uploaded to the server to update the global model. The final adversarial goal is to use the global trigger to attack the global model. Algorithms 1 and 2 illustrate the distributed backdoor attack in Federated GNNs. We first split the clients into two groups, the honest (C_h) and the malicious one (C_m) (line 2, Algorithm 1). In each round, each client updates its weights through local training (line 13, Algorithm 1), and finally, the global server aggregates local models' weights to update the global model through averaging (line 15, Algorithm 1).

The local training for every client is described in Algorithm 2. If the client is malicious

[4] Although we use the triangle as the graph trigger for each malicious client, in practice, the local triggers are more complex and different from each other.

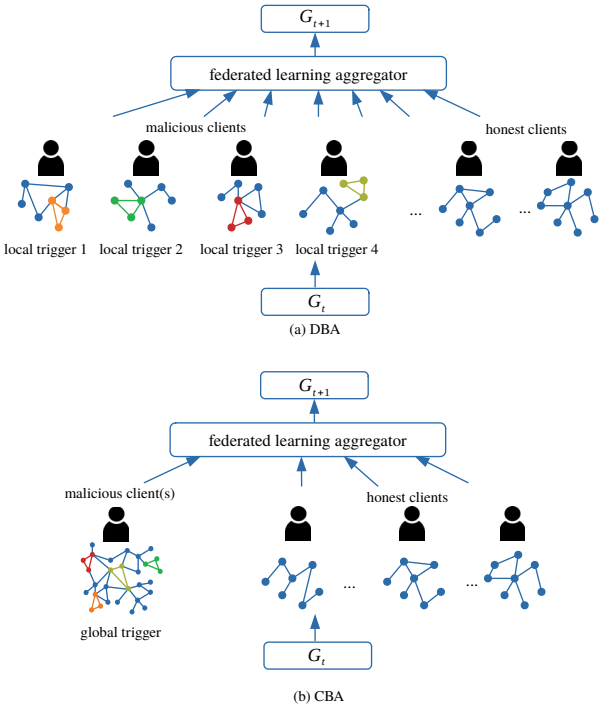


Figure 6.1: Attack Framework.

(line 2, Algorithm 2), the local training dataset will be backdoored (line 4, Algorithm 2) with the local trigger (line 3, Algorithm 2. As mentioned in Section 6.3.2, all the local triggers form the global trigger (line 5, Algorithm 2).

We conduct experiments for the malicious majority and honest majority settings to explore the impact of different percentages of malicious clients on the attack success rate. We provide additional motivation for the malicious majority setting in Section 6.8.

Centralized Backdoor Attack. Unlike DBA conducted with multiple malicious clients, CBA performs the attack with only one malicious client. CBA is a general approach in a centralized learning scenario. For example, in image classification, the attacker poisons the training dataset with a trigger so that the model misclassifies the data sample with the same trigger into the attacker-chosen label. As shown in Figure 6.1(b), the malicious client embeds its training dataset with the global trigger highlighted by four colors. This global trigger consists of local triggers used in DBA, as shown in Line 5 of Algorithm 2. Specifically, the attacker in CBA embeds its training data with four local patterns, together constituting a complete global pattern as the backdoor trigger.[5]

To compare the attack performance between the distributed backdoor attack and centralized backdoor attack in Federated GNNs, we need to make sure the trigger pattern in CBA is the union set of local trigger patterns in DBA. We can use two strategies: 1) first

[5]Here, the four colors are only used to denote four trigger patterns.

generate local triggers in DBA and then combine them to get the global trigger, or 2) first generate a global trigger in CBA and then divide it into M local triggers. We utilize the first strategy as it is an NP-hard problem to divide a graph into several subgraphs [9]. Thus, in different attack scenarios (i.e., honest majority or malicious majority attack scenarios), the CBA performance is different since the global trigger has been changed due to the different number of malicious clients.

Algorithm 1 Distributed Backdoor Attacks in Federated GNNs

Input: Dataset D, target label y_t
Output: Backdoored global model G_{t+1}, global trigger t_{global}
function DBA():
  $C_h, C_m \leftarrow ClientSplit(\text{Clients})$
  $D_{\text{local}}, D_{\text{test}} \leftarrow DataSplit(D)$
  $t_{\text{global}} \leftarrow \varnothing$
  Server executes:
  initialize G_0, $f \leftarrow$ False
  for round $t = 0, 1, 2, \ldots$ **do**
    for each client $k \in (C_h \cup C_m)$ **do**
      $w_t^k \leftarrow G_t$
      if $k \in C_m$ **then**
        $f \leftarrow$ True
      end if
      $w_{t+1}^k \leftarrow ClientUpdate(k, w_t^k, f, t_{\text{global}})$
    end for
    $G_{t+1} \leftarrow \sum_{k=1}^{K} \frac{w_{t+1}^k}{K}$
  end for
  return $G_{t+1}, t_{\text{global}}$
end function

Algorithm 2 ClientUpdate

Input: Client k, local training dataset D_{local}, current global model w, flag f, global trigger t_{global}
Output: Updated model w
function CLIENTUPDATE():
  if f is True **then**
    $t_{\text{local}} \leftarrow GenerateTrigger(s, \rho)$
    $D_{\text{local}} \leftarrow BackdoorDataset(D_{\text{local}}, t_{\text{local}}, y_t)$
    $t_{\text{global}} \leftarrow t_{\text{global}} \cup t_{\text{local}}$
  end if
  $\mathscr{B} \leftarrow$ split D_{local} into batches of size B
  for local epoch i from 1 to E **do**
    for each $b \in \mathscr{B}$ **do**
      $w \leftarrow w - \eta \nabla l(w, b)$
    end for
  end for
  return w
end function

6.4.2. BACKDOORED DATA GENERATION

We adopt the Erdős-Rényi (ER) model [15] to generate triggers (function *GenerateTrigger* in Algorithm 2) as it is more effective than the other methods (e.g., Small World model [42]

Table 6.3: Datasets statistics.

Dataset	# Graphs	Avg. # nodes	Avg. # edges	Classes	Class Distribution
NCI1	4,110	29.87	32.30	2	2,053[0], 2,057[1]
PROTEINS_full	1,113	39.06	72.82	2	663[0], 450[1]
TRIANGLES	45,000	20.85	32.74	10	4,500[0 – 9]

or Preferential Attachment model [2]) [56]. In particular, *GenerateTrigger* (line 3 in Algorithm 2), creates a random graph of s nodes. An edge between a pair of nodes in this graph is generated with probability ρ.

Backdoored data is generated (line 4 in Algorithm 2) through the following process. We sample subsets of the local training datasets (with non-target labels) with proportion r, and the rest are saved as clean datasets. For each sampled data, we inject a trigger into it by sampling s (trigger size) nodes from the graph uniformly at random and replacing their connection with that in the trigger graph. Additionally, the attacker re-labels the sampled data with an attacker-chosen target label. The backdoored data is composed of the dataset with trigger and the original clean dataset.

6.5. EXPERIMENTS

6.5.1. EXPERIMENTAL SETTING

We implemented FL algorithms using the PyTorch framework. All experiments were run on a server with 2 Intel Xeon CPUs, one NVIDIA 1080 Ti GPU with 32GB RAM. Each experiment was repeated ten times to obtain the average result. Our code is blinded for review but will be made public.

Datasets. We run experiments on three publicly available datasets: two molecular structure datasets - NCI1 [32], PROTEINS_full [6], and one synthetic dataset - TRIANGLES [24], which is a multi-class dataset. Table 6.3 provides more information about these datasets.

Dataset splits. For each dataset, we randomly sample 80% of the data instances as the training dataset and the rest as the test dataset. To simulate non-i.i.d. training data and supply each participant with an unbalanced sample from each class, we further split the training dataset into K parts following the strategy in [13] with hyperparameter 0.5 for TRIANGLES (10 classes) and hyperparameter 0.7 for other datasets (2 classes). In this paper, apart from Section 6.6 where we analyze the effect of trigger factors, we set trigger factors as follows: $\gamma = 0.2$, $\rho = 0.8$, and $r = 0.2$. As we show in Section 6.6, these hyperparameters yield an effective attack. By choosing them, we model a strong adversary that helps in evaluating the attack's behavior in the worst-case scenario.

Models and metrics. In our experiments, we use three state-of-the-art GNN models: GCN [23], GAT [39], and GraphSAGE [17].

We use the *attack success rate (ASR)* to evaluate the attack effectiveness. We embed the testing dataset with local triggers or the global trigger and then calculate the ASR of the global model on the poisoned testing dataset. We only embed the testing dataset of the non-target label with triggers to avoid the influence of the original label. The ASR measures the proportion of trigger-embedded inputs that are misclassified by the backdoored GNN into the target class y_t chosen by the adversary. The trigger-embedded

inputs are

$$D_{g_t} = \{(G_{1,g_t}, y_1), (G_{2,g_t}, y_2), \ldots, (G_{n,g_t}, y_n)\}.$$

Here, g_t is the graph trigger, $\{G_{1,g_t}, G_{2,g_t} \ldots, G_{n,g_t}\}$ is the test dataset embedded with graph trigger g_t, and $y_1, y_2, \ldots, y_n$ is the label set. Formally, ASR is defined as:

$$Attack\,Success\,Rate = \frac{\sum_{i=1}^{n} \mathbb{I}(G_{backdoor}(G_{i,g_t}) = y_t)}{n},$$

where $\mathbb{I}$ is an indicator function and $G_{backdoor}$ refers to the backdoored global model. Here, the graph trigger g_t can be local triggers or a global trigger.

6.5.2. BACKDOOR ATTACK RESULTS

We evaluate multiple-shot attack [1], which means that the attackers perform attacks in multiple rounds, and the malicious updates are accumulated to achieve a successful backdoor attack. We do not evaluate the single-shot attack [1] because the multi-shot is stealthier [34]. The multi-shot attack does not require multiplying large factors to model weights when conducting the attack, while the single-shot needs to multiply large factors to maintain the effectiveness of backdoor attacks, which can be filtered out or detected by traditional anomaly detection-based approaches such as Krum [4]. Since our main goal is conducting backdoor attacks on FL, we chose a multiple-shot attack with a high attack success rate and stealthiness. As mentioned in Section 6.3.2, we perform a complete attack in every round, showing the difference between DBA and CBA in a shorter time [46].

To explore the impact of different percentages of malicious clients on the attack performance, we evaluate the honest majority and malicious majority attack scenarios according to the percentage of malicious clients among all clients. Specifically, we set two and three malicious clients among five clients for the honest majority and malicious majority attack scenarios, respectively.

In our experiments, we evaluate the ASR of CBA and DBA with the global trigger and local triggers. The goal is to explore:

- In CBA, whether the ASR of local triggers can achieve similar performance to the global trigger even if the centralized attacker would embed a global trigger into the model.
- In DBA, whether the ASR of the global trigger is higher than all local triggers even if the global trigger never actually appears in any local training dataset, as mentioned in [46].

Honest Majority Attack Scenario. The attack results of CBA and DBA in the honest majority attack scenario are shown in Figure 6.2. Notice that the DBA ASR with a specific trigger is always higher than or at least similar to that of CBA with the corresponding trigger. For example, in Figure 6.2a (the result for the GAT model), the DBA ASR with the global trigger is higher than CBA with a global trigger. The only exception happens for GCN on TRIANGLES. We also find that the ASR of the two attacks in TRIANGLES is significantly lower than the other two datasets but still higher than random guessing. The TRIANGLES is a multi-class dataset containing complex data relations. Thus, more information needs to be encoded in each model's weights for the class features compared

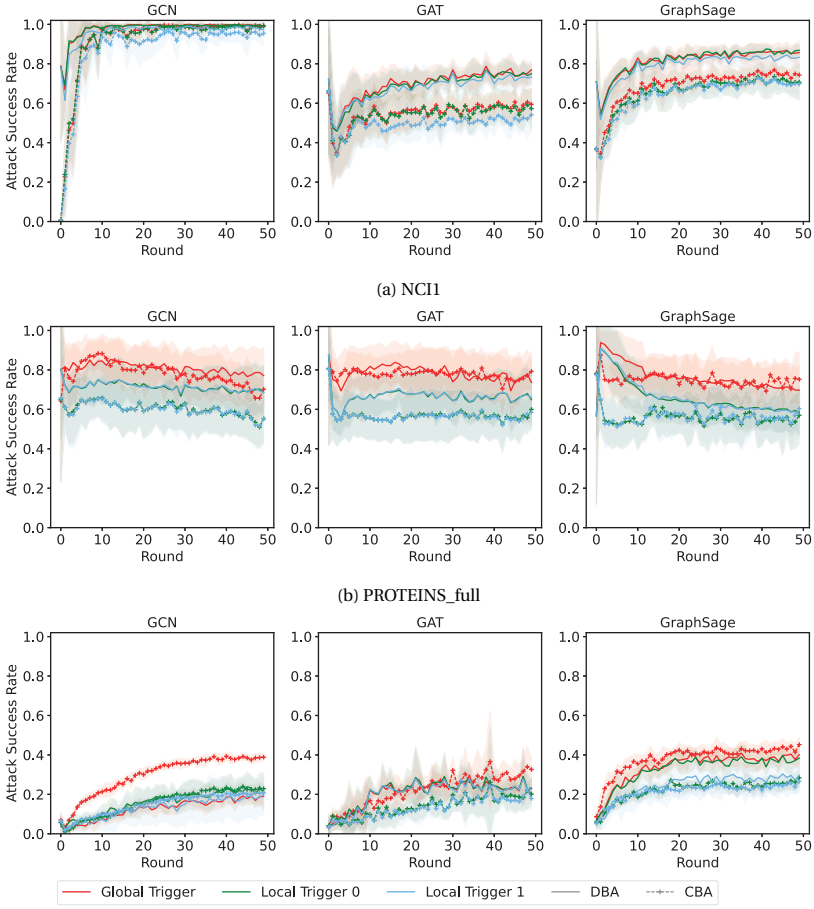
to the other datasets. As a result, there is not enough remaining space to learn our triggers easily. In most results on NCI1 and PROTEINS_full, there is an initial drop in the attack success rate for both DBA and CBA, resulting from the high local learning rate of honest clients [1]. *Based on the result for CBA, surprisingly, the ASR of all local triggers can be as high as the global trigger even if the centralized attacker embeds the global trigger into the model, which is inconsistent with the behavior in [46].* We analyze it through further experiments shown in Figure 6.8.

Moreover, the results for the PROTEINS_full dataset show that *in DBA, the attack success rate of the global trigger is higher than (or at least similar to) any local trigger, even if the global trigger never actually appears in any local training dataset.* This indicates that the high attack success rate of the global trigger does not require the same high attack success rate of local triggers. However, for the other two datasets (NCI1 and TRIANGLES), the attack success rate of the global trigger is close to all local triggers (except the result of GraphSage on TRIANGLES). This indicates that in some cases, the local trigger embedded in local models can successfully transfer to the global model so that once any local trigger is activated, the global model will misclassify the data sample into the attacker-chosen target label. This phenomenon is not consistent with the observations in [46] as in Euclidean data, most locally triggered images are similar to the clean image, but any (small) change in the structure of a graph will result in a significant dissimilarity.

Malicious Majority Attack Scenario Figure 6.3 illustrates the attack results in the malicious majority attack scenario. Compared with the honest majority attack scenario, in most cases, the attack success rate of DBA and CBA increases as with more malicious clients, more malicious updates are uploaded to the global model, making the attack more effective and persistent. Moreover, the increase in DBA is more significant than in CBA. For instance, based on the NCI1 dataset and GAT model, the DBA ASR with the global trigger in the honest majority attack scenario is 17.54% higher than CBA, while in the malicious majority attack scenario, the ASR difference is 20.65%. Thus, increasing the number of malicious clients is more beneficial for DBA than CBA. With more malicious clients, more local models are used to learn the trigger patterns in DBA, while there is only one malicious local model in CBA.

For CBA, the ASR with the global trigger is higher while the attack performance with local triggers stays at a similar level or even decreases. One possible reason is that more malicious clients mean a larger global trigger, requiring more learning capacity of the model. If there is not enough learning capacity for every local trigger in the global trigger, the backdoored model can have poor attack performance with a specific local trigger but will behave well with the union set of the local triggers, i.e., the global trigger.

Impact of the Number of Clients We only set the number of clients as 5 for these graph datasets because some of these datasets, i.e., NCI1 and PROTEINS_full, are small (less than 5,000 graphs). However, to explore the impact of the number of clients on DBA and CBA, we also conduct experiments with more clients on the largest dataset - TRIANGLES. We set the number of clients as 10 and 20 and keep the ratio of malicious clients among the total clients the same as before, i.e., 0.4 and 0.6 for the honest majority and malicious majority attack scenarios, respectively. We show the results of the honest majority attack scenario in Figure 6.4 and for the malicious majority scenario in Figure 6.5. The phenomenon between the two scenarios with 10 and 20 clients is similar to that with

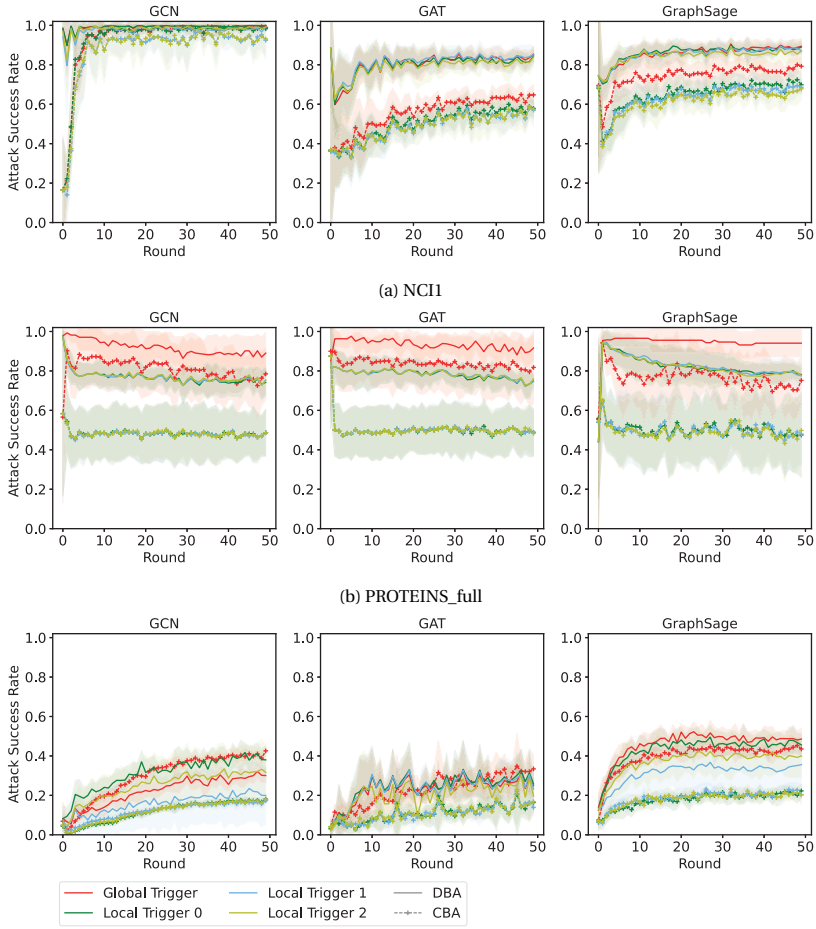


(a) NCI1

(b) PROTEINS_full

(c) TRIANGLES

Figure 6.2: Backdoor attack results in the honest majority attack scenario.



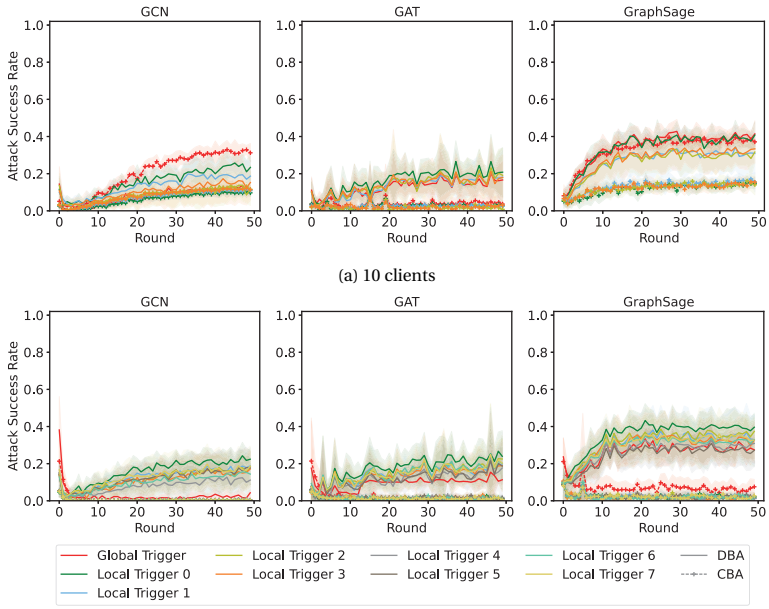
(a) NCI1

(b) PROTEINS_full

(c) TRIANGLES

Figure 6.3: Backdoor attack results in the malicious majority attack scenario.

5 clients.



(a) 10 clients

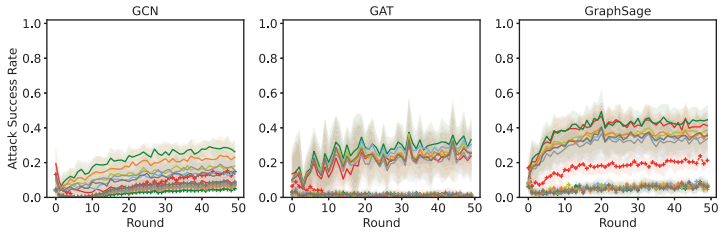
(b) 20 clients

Figure 6.4: Backdoor attack results of TRIANGLES with more clients in the honest majority attack scenario.

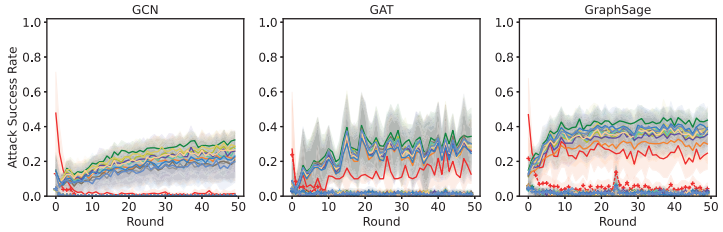
Looking at Figure 6.4, it is obvious that with the increase in the number of clients, the attack success rate of CBA decreases dramatically while the attack performance of DBA keeps steady. This is reasonable because, in CBA, there is only one malicious client whose malicious updates contribute less to the global model with more clients in total. On the contrary, in DBA, the proportion of malicious clients among total clients is the same, meaning the malicious updates contribute the same to the global model regarding the different number of clients. Therefore, as shown in Figures 6.4a and 6.4b, the number of clients has negligible impact to the DBA.

In the malicious majority attack scenario (Figure 6.5), with more clients, the DBA's ASR keeps steady while that of CBA drops dramatically, which is consistent with the observations in the honest majority attack scenario.

Impact of the Percentage of Malicious Clients Although we have analyzed the experiments with the honest majority and malicious majority scenarios, we further explore the impact of the percentage of malicious clients on the attack performance by calculating their Pearson Correlation Coefficient (PCC), as shown in Figure 6.6 (we provide the results for the GraphSage model as the example as they are more stable, and the results of other models are aligned). Recall that M represents the number of malicious clients, and each number over the line is the corresponding PCC. As we can see, for all datasets, PCC in DBA is larger than CBA, meaning the increase in M has a more positive impact on DBA than CBA. This is intuitive as more malicious clients in DBA lead to more local models embedded with local triggers, while in CBA, it means a larger global trigger due to more



(a) 10 clients



(b) 20 clients



(c) Legend

Figure 6.5: Backdoor attack results of TRIANGLES with more clients in the malicious majority attack scenario.

6

local triggers. Specifically, in DBA, more malicious clients mean more model weights to learn the trigger. In CBA, there is only one attacker, and learning a larger global trigger can be out of the model's representation capability. Additionally, as we keep the poisoning intensity of DBA and CBA the same for each malicious client, there are more poisoned training data in DBA than in CBA as more malicious clients are used.

We also explore the attack performance with more clients and less percentage of malicious clients on the large dataset - TRIANGLES. Figure 6.7 shows the attack results on TRIANGLES with 100 clients and fewer malicious clients, ranging from 5% to 20% for GraphSage, GCN, and GAT. Table 6.4 illustrates the specific attack results. We can see from Figure 6.7a that, when we use GraphSage, DBA's ASR gradually increases with more malicious clients while CBA's ASR stays below 10%, further verifying that the increase in M has a more positive impact on DBA than CBA. From Figure 6.7a, we can still observe that with 20% malicious clients, DBA can also achieve ASR of more than 20%, which means with less percentage (e.g., 20%) of malicious clients, the DBA is still effective. With more clients in total, the attack performance of CBA decreases, consistent with the observation in Figure 6.4. Similarly, for GCN (see Figure 6.7b) and GAT (see Figure 6.7c), DBA's ASR is gradually increasing with the rist in the percentage of malicious clients, and the ASR of CBA keeps steady. Thus, adding more clients does not change our previous conclusions (with 5 clients).

Analysis of CBA results In Figure 6.2, for CBA, the attack success rate of all local triggers can be as high as the global trigger, which is counterintuitive as the centralized attack only embeds the global trigger into the model. To explain these results, we further

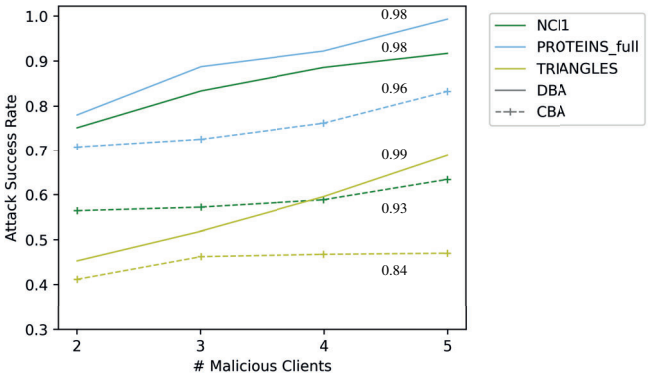
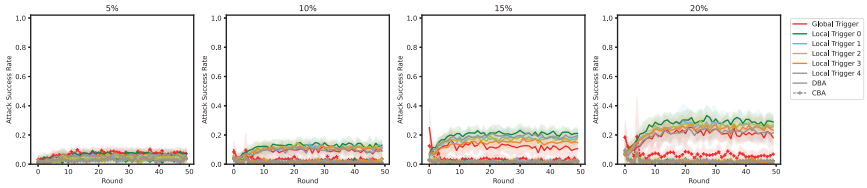
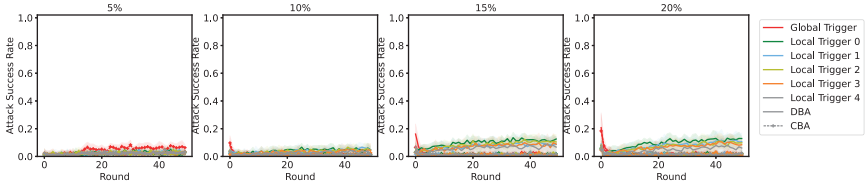


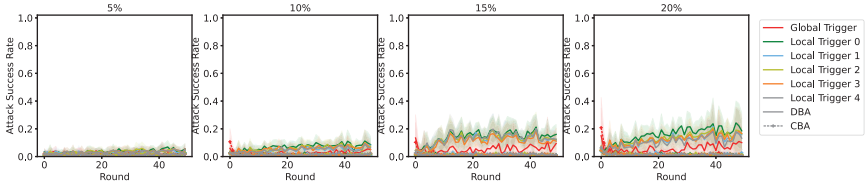
Figure 6.6: Correlation between ASR and M.



(a) GraphSage



(b) GCN



(c) GAT

Figure 6.7: Backdoor attack results of TRIANGLES with less percentage of malicious clients ($K = 100$, GraphSage, GCN and GAT).

Table 6.4: ASR of CBA and DBA with less percentage of malicious clients in TRIANGLES (K=100, GraphSage).

Model	Attack Success Rate (CBA% | DBA%)			
	5%	10%	15%	20%
GCN	6.50 | 2.03	1.36 | 1.39	1.28 | 1.06	1.52 | 1.95
GAT	1.48 | 3.91	1.11 | 3.65	1.22 | 6.97	1.51 | 8.01
GraphSage	7.64 | 6.89	2.85 | 8.80	2.87 | 10.93	5.53 | 20.49

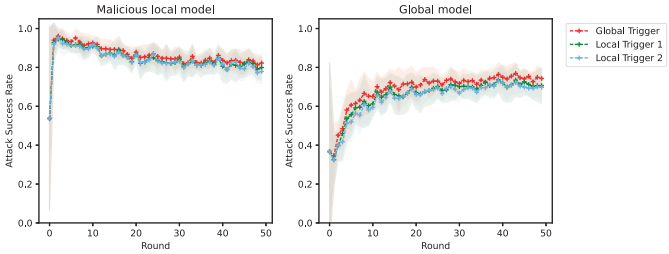
implement an experiment (NCI1 on GraphSage model) where we evaluate the attack success rate of the global trigger and local triggers in both the malicious local model [6] and the global model. As shown in Figure 6.8, in the malicious local model, the ASR of all local triggers is already close to the global trigger, which means that the malicious local model has learned the pattern of each local trigger. After aggregation, the global model inherits the capacity of local models. Once any local trigger exists, the global model will misclassify the data sample into the attacker-chosen target label.

Still, in [46], for the CBA, the attack success rate of all local triggers is significantly lower than the global trigger. There, the malicious local model learns the global trigger instead of each local trigger, so the poisoned model can only misclassify the data sample once there is a global trigger in the data. The different results in CBA between [46] and our work can be explained since there, the local triggers composing the global trigger are located close to each other (i.e., less than three pixels distance). In our work, the location of local triggers is random since a graph is non-Euclidean data where we cannot put nodes in some order. When the local trigger graphs are further away from each other, the malicious local model in CBA can only learn the local trigger instead of the global trigger.

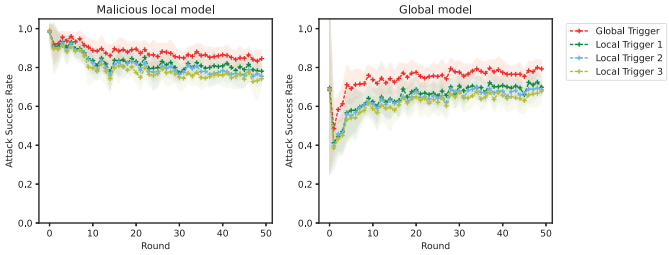
6.5.3. CLEAN ACCURACY DROP

The goal of the backdoor attack is to make the backdoored model simultaneously fit the main task and backdoor task. Therefore, it is critical that the trained model still behaves normally on untampered data samples after training with the poisoned data. Here, we use *clean accuracy drop (CAD)* to evaluate if the backdoored model can still fit the original main task. CAD is the classification accuracy difference between global models with and without malicious clients over the clean testing dataset. CBA's and DBA's final clean accuracy drop results in the honest and malicious majority attack scenarios are given in Tables 6.5 and 6.6, respectively. In most cases, both attacks have a low CAD, i.e., around 2%, and only in a few cases is there a significant CAD. These results imply that, in most cases, both attacks have a negligible impact on the original task of the model. Additionally, in some cases, DBA's CAD is significantly higher than CBA's, e.g., DBA's CAD is 5.74% in the GraphSage model on TRIANGLES while CBA's is 3.24%, as shown in Table 6.6. At the same poisoning intensity for each client, there are more poisoned data in DBA than in CBA, leading to worse performance in the main task. The substantial clean accuracy drop in DBA can also be observed in [46].

[6] For the CBA, we assume there is one centralized attacker, so there is only one local model that will be poisoned and we define this model as the malicious local model



(a) Honest majority attack scenario



(b) Malicious majority attack scenario

Figure 6.8: Centralized backdoor attack results on the malicious local model and global model with different triggers.

Table 6.5: Clean accuracy drop of CBA and DBA in the honest majority attack scenario.

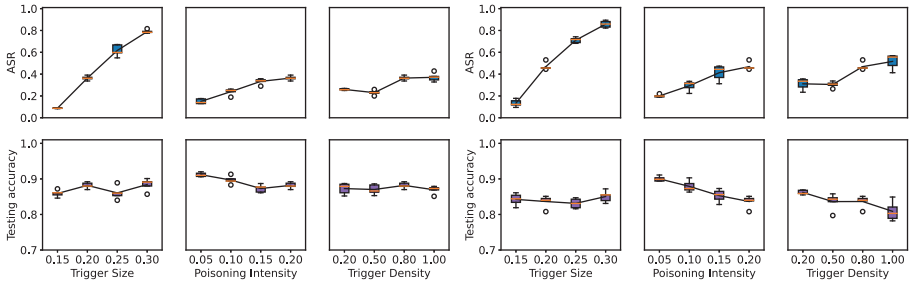
Dataset	Clean Accuracy Drop (CBA% | DBA%)		
	GCN	GAT	GraphSage
NCI1	2.54 | 2.01	0.84 | 1.42	0.93 | 0.16
PROTEINS_full	1.81 | 4.06	0.49 | 0.46	2.31 | 2.82
TRIANGLES	0.01 | 1.32	3.71 | 2.87	3.31 | 4.45

Table 6.6: Clean accuracy drop of CBA and DBA in the malicious majority attack scenario.

Dataset	Clean Accuracy Drop (CBA% | DBA%)		
	GCN	GAT	GraphSage
NCI1	4.45 | 2.74	1.03 | 1.07	1.29 | 2.22
PROTEINS_full	2.78 | 1.30	0.03 | 2.65	2.72 | 4.59
TRIANGLES	0.14 | 0.30	5.10 | 5.84	3.24 | 5.74

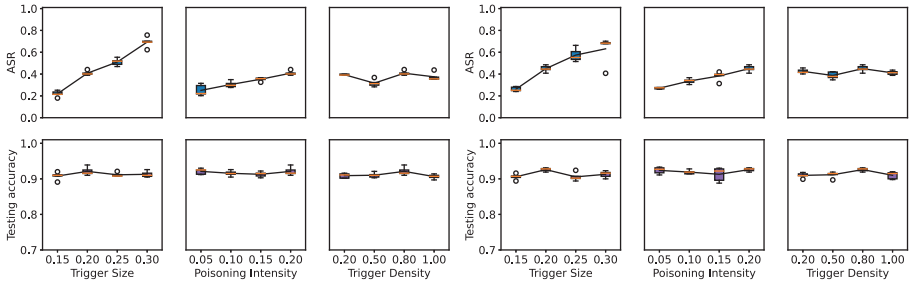
6.6. ANALYSIS OF BACKDOOR HYPERPARAMETERS

This section studies the backdoor hyperparameters discussed in Section 6.3.2. We only modify one factor for each experiment and keep other factors as in Section 6.5.1. We provide results for TRIANGLES and the GraphSage model as an example as those results are more stable, i.e., have the smallest standard error, and the results of other models are aligned. For each factor, we evaluate the global trigger's ASR and the test accuracy on the clean test dataset. We illustrate the results on TRIANGLES in two attack scenarios to analyze the effects of each factor for DBA and CBA. The results are shown in Figure 6.9.



(a) DBA in honest majority attack scenario

(b) DBA in malicious majority attack scenario



(c) CBA in honest majority attack scenario

(d) CBA in malicious majority attack scenario

Figure 6.9: Results on TRIANGLES with different trigger parameters.

Effects of Trigger Size: From the ASR results in Figure 6.9, for both attacks and attack scenarios, with the increase of trigger size, the attack success rate rises significantly, e.g., the DBA's ASR increases from 0.09 to 0.80 with trigger size rising from 0.15 to 0.30 (honest majority attack scenario). There is no significant effect of trigger size on the test accuracy of the global model, implying that the trigger size has little impact on the original main task.

Effects of Poisoning Intensity Similar to the impact of trigger size on the attack success rate, a higher poisoning intensity gives a higher attack success rate. Intuitively, a backdoor attack can perform better with more poisoned data. Nevertheless, the increase is less significant than that of different trigger sizes. Specifically, in comparison with [45], where there is no obvious difference between the impact of poisoning intensity and trigger size, here, a larger trigger size has a more positive influence on ASR than a larger poisoning intensity. We consider this an interesting observation and plan to investigate it in future work. Moreover, in DBA, the test accuracy decreases with the increasing poisoning intensity, and with more malicious clients, the drop is more significant, as shown in Figures 6.9a and 6.9b. This can be explained as with higher poisoning intensity, and more malicious clients, more model weights (including some for the original task) are influenced by the malicious trigger patterns, and the performance on the main task degrades more. We can also observe that with higher poisoning intensity, there is no obvious drop in the testing accuracy for CBA, as presented in Figures 6.9c and 6.9d. Although more local data is poisoned, the other honest clients (the majority part) still

guarantee the performance on the main task.

Effects of Trigger Density From Figure 6.9b, DBA's ASR improves from 30.10% to 47.96% when the trigger density increases from 0.50 to 0.80. This is because the average complexity of the TRIANGLES dataset is 0.16 [36]. Thus, when the trigger density is set close to this value, the difference between the original graph and the trigger graph is harder to distinguish. However, the effect of the trigger density in CBA's ASR is not strong. We see a slight fluctuation as the trigger density increases, but its range is very small to be considered a trend. In CBA, we use only one malicious client, and the weak effect of the trigger density is smoothed by the averaging operation.

In Figure 6.9, in most cases, there is no significant drop in the test accuracy with an increase in the trigger size and trigger density. On the contrary, in the backdoor attacks in centralized GNNs [48], as trigger size increases, the test accuracy decreases. This can be explained as, in FL, the influence of backdoor functionality on the main task is weakened by the aggregation of local models.

6.7. DEFENSE

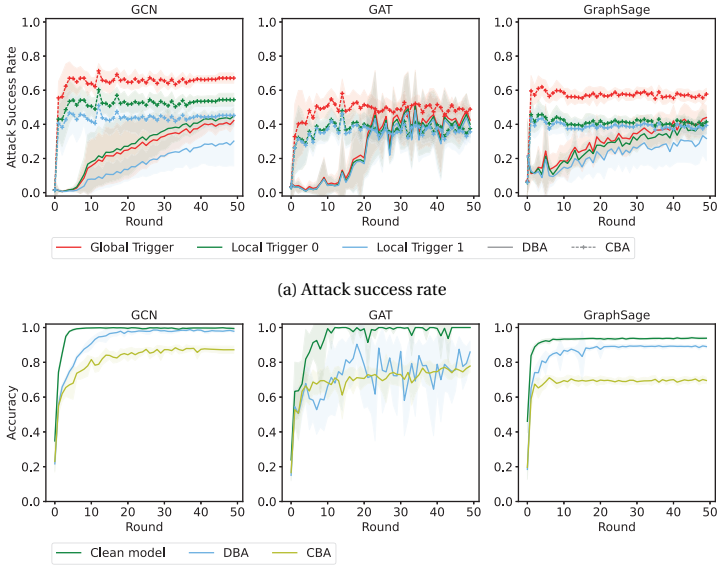
FoolsGold [14] is a robust FL aggregation algorithm that can identify attackers in federated learning based on the diversity of client updates. It reduces the aggregation weights of detected malicious clients while retaining the weights of other clients. One of the assumptions in this defense is that each client's training data is non-i.i.d and has a unique distribution, which fits the non-i.i.d data distribution setting in our paper. Thus, in this work, we focus on evaluating the attack effectiveness of DBA and CBA against FoolsGold.[7]

Figure 6.10 shows the attack performance for the TRIANGLES dataset under FoolsGold in the honest majority attack scenario (the results in the malicious majority attack scenario are similar). As illustrated in Figure 6.10a, we can observe that for DBA, the ASR under FoolsGold remains consistent for the GraphSage model, whereas it rises along with a decrease in testing accuracy for the GAT model. Moreover, generally, under FoolsGold, there is a significant increase in CBA's ASR in all models, but the testing accuracy of CBA reduces significantly at the same time. For example, the CBA's ASR increases by about 20% for the GraphSage model. However, the testing accuracy of CBA on GraphSage has a drop of more than 20% (Figure 6.10b). The results on NCI1 and PROTEINS_full (honest majority attack scenario) are shown in Figures 6.11 and 6.12, respectively. There is a slight increase in the ASR of DBA and CBA under FoolsGold, which indicates that this defense fails to identify the malicious updates and misclassifies them as benign in this case too.

Our hypothesis for this situation is that under FoolsGold, the malicious client in CBA is assigned a higher weight (recall the description of the FoolsGold mechanism from the paragraph above) than other clients, so malicious updates contribute more to the aggregated model. Simultaneously, the low weights on the honest clients' updates lead to the failure of the performance on the original task.

To verify our hypothesis, we show the FoolsGold weights on every client in the DBA and CBA using GraphSage model in Table 6.7. Here, the FoolsGold weight for each client ranges from 0 to 1. As we can see, in CBA, the weight of the malicious client is 1, and the

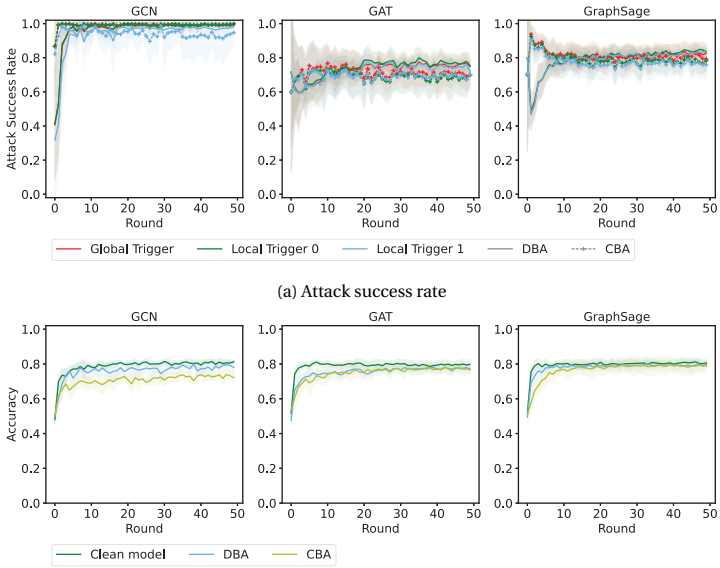
[7] In the previous version of the paper (published at the ACSAC conference [49]), there was a mistake in our implementation of FLAME indicating FLAME not to work against our attack. That statement was incorrect, and this was communicated to the authors of FLAME.



(a) Attack success rate

(b) Testing accuracy

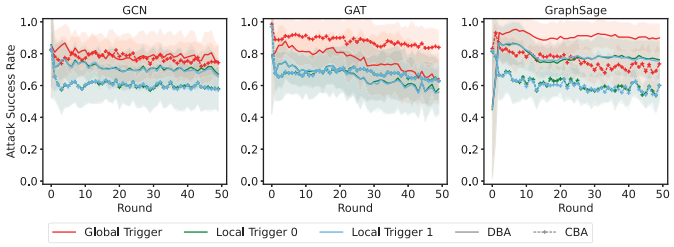
Figure 6.10: Backdoor attack results of TRIANGLES on FoolsGold for the honest majority.



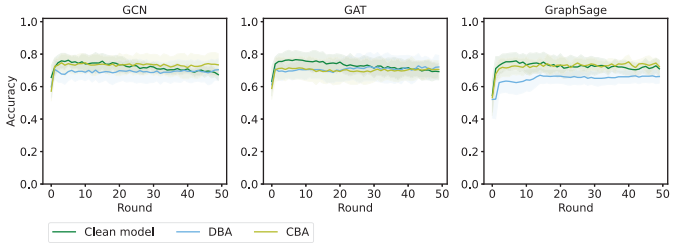
(a) Attack success rate

(b) Testing accuracy

Figure 6.11: Attack success rate on NCI1 on FoolsGold (in the honest majority attack scenario).



(a) Attack success rate



(b) Testing accuracy

Figure 6.12: Attack success rate on PROTEINS_full on FoolsGold (in the honest majority attack scenario).

weights of other clients are 0, which means only the malicious updates are aggregated into the global model. Therefore, the attack success rate of CBA increases significantly under FoolsGold. On the other hand, in DBA, the weights of the malicious clients are similar to the honest clients, indicating that the malicious and honest updates contribute equally to the aggregated model, as in the aggregation function without the defense, i.e., the average aggregation function. Therefore, there is no obvious difference between the attack performance of DBA before and after the defense.

This further verifies our hypothesis that the defense based on cosine similarity between updates is not very effective in the graph domain. The graph data are not Euclidean data, e.g., images, so the slightly different subgraphs used as triggers do not induce aligned updates. The clean accuracy drop under FoolsGold on these two datasets is similar to that without the defense. Thus, the defense does not affect the original task in that case. Additionally, in the malicious majority scenario where there are more malicious clients and the probability of detecting the malicious updates should be higher, we observe the same behavior. Thus, we conclude that the tested defense cannot detect malicious updates successfully.

Table 6.7: FoolsGold weight in DBA and CBA on TRIANGLES (honest majority attack scenario).

Attacks	Attacker 1	Attacker 2 (client 2 in CBA)	Client 3	Client 4	Client 5	Attackers (sum)
DBA	1.00 ± 0.00	1.00 ± 0.00	0.82 ± 0.15	0.81 ± 0.09	0.78 ± 0.12	2.00 ± 0.00
CBA	1.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00

6.8. RELATED WORK

Backdoor Attacks in GNNs Several recent works have conducted backdoor attacks on GNNs. Zhang et al. proposed a subgraph-based backdoor attack on GNNs for the graph classification task [56]. Xi et al. presented a subgraph-based backdoor attack on GNNs, that works for both node classification and graph classification tasks [45]. Xu et al. investigated the explainability of the impact of the trigger injecting position on the performance of backdoor attacks on GNNs and proposed a new backdoor attack strategy for the node classification task [48]. *All current attacks are implemented in centralized training for GNNs. No works explore the backdoor attacks in distributed training for GNNs, e.g., Federated GNNs.*

FL on GNNs FL has gained increasing attention as a training paradigm where data is distributed at remote devices and models are collaboratively trained in a central server. While FL has been widely studied in Euclidean data, e.g., images, texts, and sound, there are increasing studies about FL in graph data. FL on graph data was introduced in [26], where each client is regarded as a node in a graph. When it comes to detecting financial crimes (e.g., fraud or money laundering), traditional machine learning tends to lead to severe overreporting of suspicious activities. Thanks to the reasoning ability of the graph neural network, its advantages can be well-reflected. Considering the need for privacy, [38] proposed the framework for Federated GNNs to optimize the machine learning model. Besides, other research works [21, 57, 43] have been dedicated to enhancing the security of Federated GNNs. By using secure aggregation, [21] proposed a method to predict the trajectories of objects via aggregating both spatial and dynamic information without information leakage. With differential privacy, [57] and [43] put forward a framework to train Federated GNNs for vertical FL and recommendation system, respectively. Moreover, SpreadGNN was proposed in [20] to perform FL without a server. *Although there is an increasing number of works on FL for graph data, the vulnerability of Federated GNNs to backdoor attacks is still underexplored.*

The Security Assumption of Malicious Majority Clients Cao et al. took into account the situation of backdoor attacks in the malicious majority of clients and proposed a method of defense-*FLTrust* [7]. Before training begins, an honest server collects and trains on a small dataset. The server takes the updates obtained by training on a small dataset as the root of trust in each iteration. It is then compared to the updates uploaded by the clients. If the cosine similarity between them is too small, the updates will be filtered out. With this approach, the accuracy of the global model remains equivalent to that of the baseline. Based on *FLTrust*, Dong et al. considered the setting of two semi-honest servers and malicious majority clients and proposed *FLOD* to ensure that gradients are not leaked on the server side [10].

6.9. CONCLUSIONS AND FUTURE WORK

This paper explores how Centralized and Distributed Backdoor attacks behave in Federated GNNs. Through extensive experiments on three datasets and three popular GNN models, we showed that generally, DBA achieves a higher attack success rate than CBA. We showed that in CBA, the ASR of local triggers could be as high as the global trigger even if, during training, only the global trigger is embedded in the model. The impact

of the percentage of malicious clients on DBA's ASR is analyzed with correlation, where we confirm the intuition that more malicious clients lead to more successful attacks. We analyzed the critical backdoor hyperparameters to explore their impact on the attack performance and the main task. We also demonstrated that DBA and CBA are robust against one defense for the backdoor attack in FL. Interestingly, the CBA's ASR is even higher under the defense. The experimental setting in this work verifies the effectiveness of our method in a cross-silo federated learning setting and motivates further research in exploring backdoor attacks in Federated GNNs considering cross-device FL [37]. Future work will include exploring backdoor attacks in Federated GNNs for the node classification task. For example, in a social media app where each user has a local social network G^k and $\{G^k\}$ constitutes the latent entire human social network G, the developers can train a fraud detection GNN model through FL. In such a case, an attacker can conduct a backdoor attack to force the trained global model to classify a fraud node as benign.

REFERENCES

[1] Eugene Bagdasaryan et al. "How to backdoor federated learning". In: *AISTATS*. PMLR. 2020.

[2] Albert-László Barabási and Réka Albert. "Emergence of scaling in random networks". In: *science* 286.5439 (1999), pp. 509–512.

[3] Arjun Nitin Bhagoji et al. "Analyzing federated learning through an adversarial lens". In: *ICML*. PMLR. 2019.

[4] Peva Blanchard et al. "Machine learning with adversaries: Byzantine tolerant gradient descent". In: *Advances in Neural Information Processing Systems* 30 (2017).

[5] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, et al. "Towards federated learning at scale: System design". In: *arXiv preprint arXiv:1902.01046* (2019).

[6] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, et al. "Protein function prediction via graph kernels". In: *Bioinformatics* 21 (2005).

[7] Xiaoyu Cao et al. "FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping". In: *NDSS*. 2021.

[8] Dawei Cheng et al. "Financial time series forecasting with multi-modality graph neural network". In: *Pattern Recognition* (2022).

[9] Sanjoy Dasgupta, Christos H Papadimitriou, and Umesh Virkumar Vazirani. *Algorithms*. McGraw-Hill Higher Education New York, 2008.

[10] Ye Dong et al. "FLOD: Oblivious Defender for Private Byzantine-Robust Federated Learning with Dishonest-Majority". In: *Computer Security – ESORICS 2021*. Ed. by Elisa Bertino, Haya Shulman, and Michael Waidner. 2021.

[11] Shaohua Fan, Junxiong Zhu, Xiaotian Han, et al. "Metapath-guided heterogeneous graph neural network for intent recommendation". In: *Proceedings of the 25th ACM SIGKDD*. 2019.

[12] Wenqi Fan et al. "Graph neural networks for social recommendation". In: *WWW*. 2019.

[13] Minghong Fang et al. “Local model poisoning attacks to byzantine-robust federated learning”. In: *USENIX Security*. 2020.

[14] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. “Mitigating sybils in federated learning poisoning”. In: *arXiv preprint arXiv:1808.04866* (2018).

[15] E. N. Gilbert. “Random Graphs”. In: *The Annals of Mathematical Statistics* 30.4 (1959), pp. 1141–1144. DOI: 10.1214/aoms/1177706098. URL: https://doi.org/10.1214/aoms/1177706098.

[16] Zhiwei Guo and Heng Wang. “A deep graph neural network-based mechanism for social recommendations”. In: *IEEE Transactions on Industrial Informatics* 17.4 (2020), pp. 2776–2783.

[17] William L Hamilton, Rex Ying, and Jure Leskovec. “Inductive representation learning on large graphs”. In: *NeurIPS*. 2017.

[18] Andrew Hard, Kanishka Rao, Rajiv Mathews, et al. “Federated learning for mobile keyboard prediction”. In: *arXiv preprint arXiv:1811.03604* (2018).

[19] Chaoyang He et al. *FedGraphNN: A Federated Learning System and Benchmark for Graph Neural Networks*. 2021. arXiv: 2104.07145 [cs.LG].

[20] Chaoyang He et al. *SpreadGNN: Serverless Multi-task Federated Learning for Graph Neural Networks*. 2021. arXiv: 2106.02743 [cs.LG].

[21] Meng Jiang et al. “Federated dynamic gnn with secure aggregation”. In: *arXiv preprint arXiv:2009.07351* (2020).

[22] Peter Kairouz, H Brendan McMahan, Brendan Avent, et al. “Advances and open problems in federated learning”. In: *arXiv preprint arXiv:1912.04977* (2019).

[23] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: *ICLR*. 2017.

[24] Boris Knyazev, Graham W Taylor, and Mohamed Amer. “Understanding attention and generalization in graph neural networks”. In: *Advances in neural information processing systems* 32 (2019).

[25] Anusha Lalitha et al. “Peer-to-peer federated learning on graphs”. In: *arXiv preprint arXiv:1901.11173* (2019).

[26] Anusha Lalitha et al. “Peer-to-peer federated learning on graphs”. In: *arXiv preprint arXiv:1901.11173* (2019).

[27] Jaechang Lim et al. “Predicting drug–target interaction using a novel graph neural network with 3D structure-embedded graph representation”. In: *Journal of chemical information and modeling* 59.9 (2019), pp. 3981–3988.

[28] Fenglin Liu et al. “Federated learning for vision-and-language grounding problems”. In: *AAAI*. 2020.

[29] Yang Liu et al. “Fedvision: An online visual object detection platform powered by federated learning”. In: *AAAI*. 2020.

[30] Yingqi Liu et al. “Trojaning Attack on Neural Networks”. In: *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. 2018.

[31] Brendan McMahan et al. “Communication-efficient learning of deep networks from decentralized data”. In: *AISTATS*. PMLR. 2017.

[32] Christopher Morris et al. “Tudataset: A collection of benchmark datasets for learning with graphs”. In: *arXiv preprint arXiv:2007.08663* (2020).

[33] Thien Duc Nguyen et al. “Poisoning attacks on federated learning-based IoT intrusion detection system”. In: *Proc. Workshop Decentralized IoT Syst. Secur.(DISS)*. 2020.

[34] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. “Robust aggregation for federated learning”. In: *arXiv preprint arXiv:1912.13445* (2019).

[35] Omid Poursaeed et al. “Generative adversarial perturbations”. In: *CVPR*. 2018.

[36] Pavel Pudlák, Vojtěch Rödl, and Petr Savickỳ. “Graph complexity”. In: *Acta Informatica* 25.5 (1988), pp. 515–535.

[37] Virat Shejwalkar et al. “Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning”. In: *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2022, pp. 1354–1371.

[38] Toyotaro Suzumura, Yi Zhou, Natahalie Baracaldo, et al. “Towards federated graph learning for collaborative financial crimes detection”. In: *arXiv preprint arXiv:1909.12946* (2019).

[39] Petar Veličković et al. “Graph Attention Networks”. In: *ICLR* (2018). URL: https://openreview.net/forum?id=rJXMpikCZ.

[40] Daixin Wang et al. “A semi-supervised graph attentive network for financial fraud detection”. In: *ICDM*. IEEE. 2019.

[41] JIANIAN WANG et al. “A Review on Graph Neural Network Methods in Financial Applications”. In: *Journal of Data Science* 20.2 (2022), pp. 111–134.

[42] Duncan J Watts and Steven H Strogatz. “Collective dynamics of ‘small-world’networks”. In: *nature* 393.6684 (1998), pp. 440–442.

[43] Chuhan Wu et al. “Fedgnn: Federated graph neural network for privacy-preserving recommendation”. In: *arXiv preprint arXiv:2102.04925* (2021).

[44] Zonghan Wu et al. “A comprehensive survey on graph neural networks”. In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[45] Zhaohan Xi et al. “Graph backdoor”. In: *USENIX Security*. 2021.

[46] Chulin Xie et al. “Dba: Distributed backdoor attacks against federated learning”. In: *ICLR*. 2019.

[47] Zhaoping Xiong, Dingyan Wang, Xiaohong Liu, et al. “Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism”. In: *Journal of medicinal chemistry* 63.16 (2019), pp. 8749–8760.

[48] Jing Xu, Minhui Xue, and Stjepan Picek. “Explainability-based backdoor attacks against graph neural networks”. In: *Proceedings of the 3rd ACM Workshop on Wireless Security and Machine Learning*. 2021.

[49] Jing Xu et al. “More is Better (Mostly): On the Backdoor Attacks in Federated Graph Neural Networks”. In: *Proceedings of the 38th Annual Computer Security Applications Conference*. ACSAC ’22. Austin, TX, USA: Association for Computing Machinery, 2022, pp. 684–698. ISBN: 9781450397599. DOI: 10.1145/3564625.3567999. URL: https://doi.org/10.1145/3564625.3567999.

[50] Keyulu Xu et al. “How powerful are graph neural networks?” In: *arXiv preprint arXiv:1810.00826* (2018).

[51] Dong Yin et al. “Byzantine-robust distributed learning: Towards optimal statistical rates”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 5650–5659.

[52] Ruiping Yin et al. “A deeper graph neural network for recommender systems”. In: *Knowledge-Based Systems* 185 (2019), p. 105020.

[53] Rex Ying et al. “Hierarchical graph representation learning with differentiable pooling”. In: *arXiv preprint arXiv:1806.08804* (2018).

[54] Huanding Zhang et al. “Federated Graph Learning–A Position Paper”. In: *arXiv preprint arXiv:2105.11099* (2021).

[55] Muhan Zhang et al. “An end-to-end deep learning architecture for graph classification”. In: *AAAI*. 2018.

[56] Zaixi Zhang et al. “Backdoor attacks to graph neural networks”. In: *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*. 2021.

[57] Jun Zhou et al. *Vertically Federated Graph Neural Network for Privacy-Preserving Node Classification*. 2021. arXiv: 2005.11903 [cs.LG].

[58] Xinghua Zhu et al. “Empirical studies of institutional federated learning for natural language processing”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*. 2020.

7

CatBack: Universal Backdoor Attacks on Tabular Data via Categorical Encoding

Backdoor attacks in machine learning have drawn significant attention for their potential to compromise models stealthily, yet most research has focused on homogeneous data such as images. In this work, we propose a novel backdoor attack on tabular data, which is particularly challenging due to the presence of both numerical and categorical features. Our key idea is a novel technique to convert categorical values into floating-point representations. This approach preserves enough information to maintain clean-model accuracy compared to traditional methods like one-hot or ordinal encoding. By doing this, we create a gradient-based universal perturbation that applies to all features, including categorical ones.

We evaluate our method on five datasets and four popular models. Our results show up to a 100% attack success rate in both white-box and black-box settings (including real-world applications like Vertex AI), revealing a severe vulnerability for tabular data. Our method is shown to surpass the previous works like Tabdoor in terms of performance, while remaining stealthy against state-of-the-art defense mechanisms. We evaluate our attack against Spectral Signatures, Neural Cleanse, Beatrix, and Fine-Pruning, all of which fail to defend successfully against it. We also verify that our attack successfully bypasses popular outlier detection mechanisms.

This chapter has been published as B. Tajalli, S. Koffas, S. Picek, (2026 February). "CatBack: Universal Backdoor Attacks on Tabular Data via Categorical Encoding." *In Proceedings of the 33st Annual Network and Distributed System Security Symposium (NDSS) (Vol. 2026).*

7.1. Introduction

Machine learning on tabular data represents a state-of-the-art real-world approach for applications like healthcare analytics [43], credit risk assessment [11], and fraud detection [6]. Unlike image or text data, tabular data often contains a mix of numerical and categorical features, which poses unique challenges in preprocessing and model design. Indeed, despite the rising popularity of specialized deep learning models for tabular data [18, 39], traditional methods like gradient boosting machines remain highly effective and widely used in industry and research [19].

The wide adoption of machine learning systems led to diverse security threats [30], where most research focuses on attacks on computer vision systems. One such threat is the backdoor attack, a powerful attack where an adversary injects a hidden trigger into the training set, causing the model to misclassify specific inputs at test time [29, 40]. Despite the vast amount of research related to backdoor attacks, only a few works consider backdoor attacks for tabular data [48, 24, 36, 35]. Tabular data must be handled differently from other data types, as they are heterogeneous and use dense numerical or sparse categorical features [5], which complicates the trigger design. For this reason, existing backdoor attacks mostly use only numerical features [36, 48] for the trigger, which limits the attacker's power as real-world tables frequently have both feature types. Additionally, unlike speech and image data, the correlation between features is weak. Thus, there is no spatial information that the attacker could exploit [5] to make an effective backdoor trigger. Moreover, previous works typically rely on encoding methods such as one-hot transformations, which do not allow the attacker to craft the trigger as freely as it does in the image domain. Consequently, previous works used heuristic approaches [36, 48, 35], as, though not impossible, optimization-based approaches become more challenging.

This paper addresses the challenge of backdoor attacks on tabular data containing numerical and categorical columns. We propose a novel conversion method that turns categorical columns into floating-point representations, enabling a unified feature space where every dimension can be targeted by a single, gradient-based perturbation. Notably, this conversion can compete with, or even surpass, popular encodings like one-hot and ordinal methods regarding clean accuracy. It also simplifies model training by eliminating the need to explicitly define categorical columns and their number of categories for transformer-based models. Leveraging this encoding, we craft a *universal backdoor perturbation*, which can be applied to any input data type, forcing it to have a new value (thus not a universal fixed trigger for all inputs). This method achieves up to 100% attack success rate in both white-box and black-box scenarios. We evaluated our attack using four state-of-the-art models and five benchmark tabular datasets. To underscore the real-world feasibility and impact of our backdoor attack, we deployed it against Google AutoML [16], a prominent commercial machine learning service. The attack remained effective, highlighting serious practical risks even in state-of-the-art industrial systems. We also conducted a comprehensive evaluation of our attack on benchmark defensive measures and outlier detection methods. Our findings highlight a serious vulnerability for tabular data, stressing the need for more robust defenses in security-sensitive settings.

Our main contributions are as follows:

- We introduce a novel backdoor on tabular data called CatBack that uses any combination of feature types for its trigger. In particular, our encoding of categorical

features enables the attacker to apply a universal perturbation to any data type, providing more freedom to create stealthier triggers.
- We apply our attack on five datasets and four models (both neural networks and classical machine learning methods), showing that it can generalize well in different settings. In particular, our attack reached ≈ 100% attack success rate in most cases.[1] For example, our attack outperformed two baseline attacks (Badnets [20] and Tabdoor [36]) by up to 44% and 95% on the ACI dataset respectively and by up to 16% and 71% on the BM dataset.
- We evaluated our method against state-of-the-art defenses, including Spectral Signatures, Neural Cleanse, Beatrix, and Fine-Pruning. Our results demonstrate that in most cases, CatBack bypasses these defenses. For instance, even under the least favorable settings for the attacker, CatBack still managed to evade Spectral Signature in over 78% of cases. For Fine-Pruning, more than 75% of results fail to mitigate the attack, and for Neural Cleanse and Beatrix, all of the attacker's attempts go undetected.
- We introduce a novel way to encode categorical features in tabular data by converting them to real numbers. Using our method, the models can achieve the same performance as common benchmark methods. In our encoding, there is no need for extended columns for each feature, such as one-hot encoding or embedding layers, as the values are treated the same as real numbers.
- We verified that our attack works on a widely used platform (Vertex AI), showing that such attacks can happen in real-world scenarios.

7.2. BACKGROUND

7.2.1. BACKDOOR ATTACKS

Backdoor attacks insert a hidden "trigger" into a model during training so that at inference time, only inputs carrying that trigger will be misclassified into an adversary's chosen label. Formally, let

$$\mathscr{F}_\theta : \mathscr{X} \to \mathscr{Y}.$$

be a neural network with parameters θ, mapping feature space $\mathscr{X}$ to labels $\mathscr{Y}$. A trigger function $\mathscr{T} : \mathscr{X} \to \mathscr{X}$ embeds a pattern δ into any clean sample $\mathbf{x}$, such that

$$\mathscr{F}_\theta\big(\mathscr{T}(\mathbf{x})\big) = y_t \quad \forall \mathbf{x} \in \mathscr{X},$$

where $y_t \in \mathscr{Y}$ is the target class the attacker chooses.

Backdoors can be introduced in several ways:

1. **Data Poisoning:** Insert m poisoned samples $\{(\hat{\mathbf{x}}_j, \hat{y}_j)\}_{j=1}^m$ into the clean set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, so that the poisoning rate $\epsilon = m/n$ stays small [20, 29].
2. **Code Poisoning:** Modify the training pipeline or loader to apply $\mathscr{T}$ on a subset of input data [4].
3. **Model Poisoning:** Directly alter learned parameters θ after training to respond to δ [23].

[1] Our code is available at https://github.com/catback-tabular/catback.git.

Focusing on data poisoning, the attacker's training objective becomes

$$\theta^* = \arg\min_{\theta} \Big[\sum_{i=1}^{n-m} \mathscr{L}\big(\mathscr{F}_\theta(\mathbf{x}_i), y_i\big) + \sum_{j=1}^{m} \mathscr{L}\big(\mathscr{F}_\theta(\hat{\mathbf{x}}_j), \hat{y}_j\big) \Big],$$

where $\mathscr{L}$ is, e.g., cross-entropy loss. After training, the model should satisfy

$$\mathscr{F}_{\theta^*}(\mathbf{x}) \approx \mathscr{F}_\theta(\mathbf{x}) \quad \forall \mathbf{x} \in D_{\text{clean}},$$
$$\mathscr{F}_{\theta^*}(\mathscr{T}(\mathbf{x})) = y_t \quad \forall \mathbf{x} \in \mathscr{X}.$$

A successful attack adheres to the following assumptions:
1. ϵ should be very small so poisoned points blend in.
2. Clean-data accuracy must stay nearly unchanged, i.e., $\mathscr{F}_{\theta^*}(\mathbf{x}) \approx \mathscr{F}_\theta(\mathbf{x})$ for clean $\mathbf{x}$ [1].

7.2.2. Characteristics of Tabular Data

Tabular data consists of heterogeneous features that may follow varying data types and distributions [5]. Thus, creating triggers for tabular data requires a different approach than domains like images or text. As discussed in [36], tabular data has the following properties that may affect the design of the backdoor trigger:

- **Data Heterogeneity**: Each feature may follow a distinct distribution, so designing a universal trigger is not straightforward. Additionally, some features may take a specific range of values, and anything outside this range may be easily classified as an outlier, unlike the pixels in an image.
- **Mutually Exclusive Features**: Categorical features are often encoded in representations like one-hot encoding (OHE). Such representations do not allow unrestricted perturbation, as any trigger that activates multiple values could be easily identified as an outlier and removed from the dataset.
- **Absence of Spatial Relationships**: In tabular data, there is no notion of spatial or sequential dependency among the features. Thus, embedding a trigger into different features does not propagate naturally like the one in images or text. Additionally, the order of the features is unimportant in tabular data.
- **Prediction Sensitivity on Important Features**: In tabular data, some features may affect the model's decision significantly more than others, complicating the design of stealthy backdoor triggers.

7.2.3. Machine Learning Models for Tabular Data

A tabular dataset is defined as

$$\mathscr{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}, \quad \mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{id}]^\top,$$

where each sample $\mathbf{x}_i$ has d features that may be continuous or categorical [5]. Traditional methods like decision trees, random forests, and gradient boosting (e.g., XGBoost [7], LightGBM [25]) remain strong baselines because they handle mixed feature types and missing values naturally.

Beyond these, several architectures have been tailored for tabular data [3, 39, 18]:

1. **Hybrid Models:** Combine tree-based splits or feature embeddings with dense layers to capture both simple rules and complex interactions (e.g., NODE, DeepFM).
2. **Transformer-based Models:** Use self-attention to learn pairwise feature dependencies. Formally:

$$\mathscr{F}_\theta(\mathbf{x}) = \text{Transformer}(\mathbf{x};\theta), \tag{7.1}$$

$$\text{Attention}(Q,K,V) = \text{softmax}\Big(\frac{QK^\top}{\sqrt{d_k}}\Big)V. \tag{7.2}$$

In this work, we evaluate both kinds of models on our tabular benchmarks. We picked the top leading models from previous studies [5, 19, 18] which are also widely incorporated in industrial platforms [2]:

- *Classical:* XGBoost [7].
- *Deep:* TabNet [3], Saint [39], FT-Transformer (FTT) [18].

Using both options ensures we cover established ensemble methods and recent deep models.

All models learn a mapping:

$$\mathscr{F}_\theta : \mathbb{R}^d \to \mathscr{Y},$$

but differ in how they preprocess features, handle sparsity, and capture interactions. This variety helps us assess backdoor vulnerability across a wide range of tabular learners.

7.3. THREAT MODEL

In this work, we introduce a novel backdoor attack targeting neural network models trained on tabular data, with a particular emphasis on manipulating categorical features - a direction mostly neglected in prior research. Indeed, previous attacks predominantly focused on numerical columns, modifying them to implant backdoors [48, 36]. Our approach extends this paradigm by incorporating categorical features into the attack vector, thereby enhancing the potential effectiveness and stealthiness of the backdoor. We consider a classification task where a neural network model $F : \mathscr{X} \to \mathscr{Y}$ maps inputs from the feature space $\mathscr{X} \subseteq \mathbb{R}^d$ to the label space $\mathscr{Y} = \{1, 2, \ldots, C\}$, where C is the number of classes.

7.3.1. ATTACKER'S CAPABILITIES AND OBJECTIVES

We assume an attacker with full access to the training dataset $D_{original} = \{(x_i, y_i)\}_{i=1}^N$. The attacker's objective is to implant a dirty label backdoor into the model such that any input modified with a specific trigger pattern will cause the model to predict a target label $t \in \mathscr{Y}$, irrespective of the input's true label. We apply our threat model to two main scenarios:

- White box: The attacker has full knowledge of the victim model and may or may not control the training process. This scenario can be applied in outsourced training.
- Black box: The attacker does not know the architecture of the victim model and has no control over the training process. This scenario is realistic in the context of dataset poisoning.

7.4. CONVERTING CATEGORICAL FEATURES

By employing the frequency mappings from [13], we implement a hierarchical mapping strategy with an adaptive tie-breaker Δr (see Section 7.4.2) that ensures a unique numerical representation for each category within a categorical feature. We call this mapping function $Conv(.)$ where $D = Conv(D_{original})$.

7.4.1. PRIMARY FREQUENCY-BASED MAPPING

For each categorical feature j with unique values $\mathcal{V}_j = \{v_{j1}, v_{j2}, \ldots, v_{jk_j}\}$, where k_j is the number of categories, perform the following:

1. **Compute Frequencies:** Calculate the frequency c_{jl} of each category v_{jl} in the dataset $D_{original}$.
2. **Initial Mapping:** Assign r_{jl} using the formula:

$$r_{jl} = \frac{c_{\max,j} - c_{jl}}{c_{\max,j} - 1}, \quad \text{for} \quad l = 1, \ldots, k_j,$$

where $c_{\max,j} = \max_{1 \le l \le k_j} c_{jl}$.

As discussed in [13], this frequency-based transformation creates values in the interval of $[0,1]$. The most frequent value is mapped to 0, and the least frequent values are closer to 1. In the extreme case that the least frequent value has a frequency of 1, then $r_{ij} = \frac{c_{max,j}-1}{c_{max,j}-1} = 1$.

7.4.2. ADAPTIVE Δr SELECTION

Some categories may share the same frequency, which leads to a tie during conversion. To determine Δr precisely and avoid the tie, we follow an approach based on the smallest decimal precision in the primary mapping.

1. **Sort Unique r_{jl} Values:** Sort the unique r_{jl} values in ascending order.
2. **Compute Minimum Difference:**

$$\Delta r_{\min} = \min_i \left(r_{jl}^{(i+1)} - r_{jl}^{(i)} \right).$$

3. **Determine p:** Identify the largest single decimal component in $\Delta r_{\min}$. Specifically, express $\Delta r_{\min}$ in decimal form and determine the smallest decimal place p where a non-zero digit occurs.
   - **Definition of p:** Let $\Delta r_{\min} = 0.d_1 d_2 \ldots d_n$, where d_1 is the first non-zero digit. Then, p is the position of the first non-zero digit.
4. **Set Δr:** Define Δr as:

$$\Delta r = 10^{-(p+1)}.$$

This ensures that Δr is one order of magnitude smaller than the smallest decimal precision in $\Delta r_{\min}$, maintaining uniqueness without overlapping existing r_{jl} values.

- **Example:**
  - If $\Delta r_{\min} = 0.4$ (first decimal place), then $p = 1$ and $\Delta r = 0.01$.
  - If $\Delta r_{\min} = 0.04$ (second decimal place), then $p = 2$ and $\Delta r = 0.001$.

7.4.3. IDENTIFYING AND RESOLVING TIES

1. **Detect Tied Categories:** For each feature j, identify sets of categories that share the same frequency c_{jl}, as each category should have a unique value. This is necessary to ensure that our mapping table is reversible and that the model can distinguish between different categories during training.
2. **Apply Secondary Ordering:** For each set of tied categories, apply a deterministic secondary ordering criterion, such as alphabetical order.
3. **Assign Unique Offsets:** For each category v_{jl} in a tied set, assign a unique r'_{jl} by adding incremental multiples of Δr based on the secondary order:

$$r'_{jl} = r_{jl} + (k-1) \times \Delta r,$$

where k is the position in the secondary ordering (starting from 1).

7.4.4. FINAL NUMERICAL REPRESENTATION

The final numerical representation for each category v_{jl} is:

$$r'_{jl} = \begin{cases} r_{jl} + (k-1) \times \Delta r, & \text{if } v_{jl} \text{ is part of a tied set} \\ r_{jl}, & \text{otherwise,} \end{cases}$$

ensuring that each category has a unique r'_{jl} value.[2]

7.4.5. REVERSE MAPPING

To facilitate efficient reverse mapping from numerical values r'_{jl} back to their original categorical values v_{jl}, we implement a structured lookup mechanism. The process involves the following steps:

1. **Construction of the Lookup Table:** During the encoding phase, alongside assigning each category its unique numerical representation r'_{jl}, we construct a lookup table T_j for each categorical feature j. The table T_j maps each r'_{jl} to its corresponding category v_{jl}:

$$T_j = \{(r'_{jl}, v_{jl}) \mid v_{jl} \in \mathcal{V}_j\}.$$

.

This table can be efficiently implemented using data structures such as hash tables or dictionaries, enabling constant-time $O(1)$ access during reverse mapping, and minimal memory requirements, as hash tables have $O(n)$ space complexity, where n is the number of categories each feature has.

2. **Reverse Mapping Function:** To retrieve the original category from a given r'_{jl}, the reverse mapping function performs the following:
   - **Lookup Operation:** Given an r'_{jl}, query the lookup table T_j to obtain the corresponding category v_{jl}.

[2] Although unlikely, it is (theoretically) possible that a new tie occurs after the current tie resolution is performed. In such a case, Steps 7.4.2 and 7.4.3 of the algorithm must be repeated with new r'_{jl} values until no new ties exist anymore.

- **Handling Precision:** Ensure that the r'_{jl} values used during the attack or optimization process are matched exactly to those stored in T_j. Implement rounding mechanisms if necessary to align floating-point representations.

Formally, the reverse mapping function $RevConv(r'_{jl})$ is defined as:

$$RevConv(r'_{jl}) = v_{jl} \quad \text{such that} \quad (r'_{jl}, v_{jl}) \in T_j.$$

We also define the function $Revert(.)$ that reverts the whole dataset from converted numerical values to categorical values again.

7.4.6. EXAMPLE OF OUR ENCODING AND ATTACK ALGORITHM

Here, we demonstrate our encoding method through an example. To this end, we consider a categorical feature j with the following category counts:

Category v_{jl}	**Count** c_{jl}
A	50
B	50
C	30
D	20

Table 7.1: Category Counts for Feature j

Primary Mapping:

$$r_{jA} = \tfrac{50-50}{50-1} = 0.0000, \qquad r_{jB} = \tfrac{50-50}{50-1} = 0.0000$$
$$r_{jC} = \tfrac{50-30}{50-1} \approx 0.4082, \qquad r_{jD} = \tfrac{50-20}{50-1} \approx 0.6122$$

Determine Δr:

- $\Delta r_{\min} = 0.204$
- Identify the largest single decimal component in $\Delta r_{\min}$:
  - $\Delta r_{\min} = 0.204$ has the first non-zero decimal at the first decimal place ($p = 1$).
- Set $\Delta r = 10^{-(p+1)} = 10^{-2} = 0.01$.

Apply Hierarchical Mapping:

$$r'_{jA} = 0.0000 + (1-1) \times 0.01 = 0.0000$$
$$r'_{jB} = 0.0000 + (2-1) \times 0.01 = 0.0100$$
$$r'_{jC} = 0.4082$$
$$r'_{jD} = 0.6122$$

Result:

Lookup Table:

Referring to Table 7.2, the lookup table T_j for feature j is constructed as:

When an r'_{jl} value of 0.0100 is encountered during reverse mapping, the corresponding category retrieved from T_j is B.

Category v_{jl}	Original r_{jl}	Updated r'_{jl}
A	0.0000	0.0000
B	0.0000	0.0100
C	0.4082	0.4082
D	0.6122	0.6122

Table 7.2: Updated Numerical Representation After Hierarchical Mapping

Numerical Value r'_{jl}	Category v_{jl}
0.0000	A
0.0100	B
0.4082	C
0.6122	D

Table 7.3: Lookup Table T_j for Feature j

7.5. ATTACK METHODOLOGY

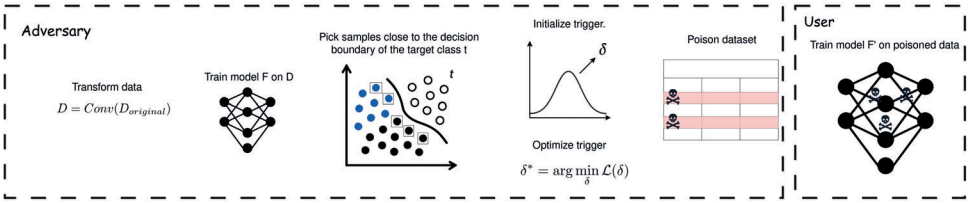


Figure 7.1: CatBack's schematic. Initially, the attacker (using our encoding) trains a model F on the transformed dataset. Then, the attacker selects samples close to the decision boundary of the target class t. Starting from a trigger randomly initialized with the appropriate values (retrieved from a normal distribution), the attacker poisons a fraction of the data of the used dataset. Finally, the attacker reverts the dataset to its correct state so users can use it to train their own models.

In Figure 7.1, we depict a schematic of the CatBack attack. Next, we discuss each of the attack steps in detail.

7.5.1. INITIAL MODEL TRAINING

The dataset's categorical features are transformed to numerical using the method explained in Section 7.4. Then, the attacker trains the model F on the converted dataset $D = Conv(D_{original})$ to obtain a baseline model that performs adequately on the classification task.

7.5.2. SELECTION OF NON-TARGET SAMPLES

The attacker constructs a subset $D_{\text{non-target}}$ by excluding all samples with the target label t:

$$D_{\text{non-target}} = \{(x_i, y_i) \in D \mid y_i \neq t\}.$$

7.5.3. CONFIDENCE-BASED SAMPLE RANKING

The attacker evaluates the trained model F on $D_{\text{non-target}}$ to obtain the softmax confidence scores for the target class t. For each input x_i, the confidence score equals

$$s_i = f_t(x_i),$$

where $f_t(x_i)$ is the softmax output corresponding to class t.

The attacker pairs each input with its confidence score to form the set:

$$D_{\text{conf}} = \{(x_i, s_i) \mid (x_i, y_i) \in D_{\text{non-target}}\}.$$

The attacker then sorts D_{conf} in descending order based on s_i and selects the top $\mu \cdot |D_{\text{conf}}|$ samples to create the subset D_{picked}, where $\mu \in (0, 1]$ is a predefined fraction (e.g., $\mu = 0.2$). By doing this, we want to find the samples closer to the target decision boundary. This helps us to craft a minimal perturbation. The lower μ value means we have chosen the non-target samples closer to the target class; thus, values calculated for the perturbation vector would decrease. On the other hand, increasing the μ invites more samples into consideration, making the perturbation vector more general and with higher values.

7.5.4. DEFINITION OF THE BACKDOOR TRIGGER

The attacker defines a universal trigger pattern $\delta \in \mathbb{R}^d$ to be added to the inputs. More specifically, δ is a randomly initialized sample of the dataset.[3] The backdoored input $\hat{x}_i$ is computed as:

$$\hat{x}_i = \text{clip}(x_i + \delta),$$

where the clipping function ensures that each feature of $\hat{x}_i$ remains within its valid range:

$$\hat{x}_i^{(j)} = \begin{cases} \max X^{(j)}, & \text{if } x_i^{(j)} + \delta^{(j)} > \max X^{(j)}, \\ \min X^{(j)}, & \text{if } x_i^{(j)} + \delta^{(j)} < \min X^{(j)}, \\ x_i^{(j)} + \delta^{(j)}, & \text{otherwise,} \end{cases}$$

with $\min X^{(j)}$ and $\max X^{(j)}$ being the minimum and maximum values of feature j in D. This makes the final perturbed sample instance dependent. Thus, the resulting backdoored sample is unique to each original sample, while the perturbation vector δ is universal.

7.5.5. OPTIMIZATION OF THE TRIGGER PATTERN

The attacker optimizes δ by minimizing the following loss function over D_{picked}:

$$\mathscr{L}(\delta) = \frac{1}{|D_{\text{picked}}|} \sum_{(x_i, y_i) \in D_{\text{picked}}} \Big[-\log f_t(\hat{x}_i) + \beta \|\hat{x}_i - \text{Mode}(X)\|_1 + \lambda \|\hat{x}_i - \text{Mode}(X)\|_2^2 \Big],$$

where:

[3] In our experiments, we initialized δ to 0.

- $f_t(\hat{x}_i)$ is the softmax output for class t given input $\hat{x}_i$.
- $\text{Mode}(X) \in \mathbb{R}^d$ is the mode vector of the dataset D, with each element $\text{Mode}(X)^{(j)}$ being the empirical mode value[4] of feature j.
- β and λ are hyperparameters controlling the L_1 and L_2 regularization terms, respectively.

The loss function balances two objectives:

1. Maximizing the model's confidence in predicting the target class t for the backdoored inputs.
2. Ensuring the trigger pattern δ keeps the modified inputs close to common data patterns (via the mode value) to enhance stealthiness.

To craft a trigger that is both sparse (few features changed) and stable (no extreme perturbations), we combine L_1 and L_2 penalties in the loss. The L_1 term $\|\hat{x}_i - \text{Mode}(X)\|_1$ drives some coordinates of the trigger δ to zero, limiting the number of features that are modified [41]. The L_2 term $\|\hat{x}_i - \text{Mode}(X)\|_2^2$ prevents the nonzero perturbations from growing too large, yielding smoother gradients and better conditioning [22]. Together, this "elastic-net" style regularization has been shown to outperform pure L_1 or L_2 alone—especially when features are correlated—by offering both reliable feature selection and numerical stability [51].

The optimized trigger pattern δ^* is obtained by solving:

$$\delta^* = \arg\min_{\delta} \mathscr{L}(\delta).$$

This optimization is performed using gradient descent, updating δ iteratively based on the gradient $\nabla_\delta \mathscr{L}$.

After the optimization process, where r'_{jl} values might be adjusted continuously, it is crucial to maintain valid categorical representations. This is achieved by:

- **Rounding Adjusted Values:** Any continuous changes to r'_{jl} are rounded to the nearest valid value present in the lookup table T_j:

$$r'^{\text{rounded}}_{jl} = \text{round}(r'_{jl}, \text{precision} = p').$$

  where p' corresponds to the decimal precision used in Δr.
- **Validation:** Ensure that the rounded r'^{rounded}_{jl} exists within T_j. If not, adjust r'_{jl} to the closest valid value to maintain consistency.

7.5.6. CONSTRUCTION OF THE POISONED DATASET

With δ^* optimized, the attacker selects randomly a fraction $\epsilon \in (0, 1]$ of the dataset D to poison. In other words, ϵ is the poisoning rate of our attack.

Each selected sample (x_i, y_i) is modified:

$$\hat{x}_i = \text{clip}(x_i + \delta^*), \quad \hat{y}_i = t.$$

The poisoned dataset D_{poisoned} consists of the modified samples:

$$D_{\text{poisoned}} = \{(\hat{x}_i, \hat{y}_i) \mid (x_i, y_i) \in D_{\text{selected}}\},$$

[4] The mode of a distribution is defined as the most frequent value in a given dataset or probability distribution. We used `pandas.DataFrame.mode` for this task.

where $D_{\text{selected}} \subset D$ and $|D_{\text{selected}}| = \epsilon \cdot N$.

Finally, we revert the data to their original form through our $Revert(\cdot)$ function, so that the final training dataset is:

$$D' = Revert((D \setminus D_{\text{selected}}) \cup D_{\text{poisoned}}).$$

7.5.7. Training the Backdoored Model

When the poisoned trainset is ready, the attacker can retrain the model F' on the poisoned trainset D' or upload the trainset to some public node and have the user do the training.

Note that to handle categorical features in D', we adhere to standard preprocessing protocols that an innocent user would typically employ [48, 3]. This includes utilizing encoding techniques such as embedding methods (with random [12] or Xavier [14] initialization) or one-hot encoding to transform categorical variables into a numerical format suitable for neural network training. In this study, we employ the `OrdinalEncoder` from scikit-learn to convert all categorical features into ordinal integers.

The expectation is that F' maintains performance on clean data while exhibiting the backdoor behavior when the trigger is present, so that the user will not become suspicious about the model. In this way, the model will classify "clean" samples correctly without raising any suspicions, but will be manipulated by the attacker through "poisoned" inputs.

7.5.8. Deployment and Attack Activation

During deployment, any input x modified with the trigger pattern δ^* will be misclassified as the target label t:

$$F'(\hat{x}) = F'(x + \delta^*) = t.$$

Algorithm 3 summarizes our attack steps.

7.5.9. Hyperparameter Considerations

The hyperparameters μ, β, λ, and ϵ play crucial role in the attack:

- μ ($0 < \mu \leq 1$) controls the proportion of high-confidence samples used for optimizing δ. A higher μ may lead to a more generalized trigger but could also increase optimization difficulty.
- As described in Section 7.5.5, β and λ ($\beta, \lambda > 0$) control the magnitude of perturbation and number of affected features, which regulate the stealthiness and attack efficacy. They should be set to balance minimizing perturbations and maximizing the model's confidence in the target class.
- ϵ ($0 < \epsilon \leq 1$) determines the fraction of the dataset to poison. A smaller ϵ enhances stealth but may reduce the backdoor's effectiveness.

These hyperparameters can be tuned based on experimental results to achieve the desired trade-offs between attack success rate, stealthiness, and impact on model performance. We have tuned β and λ through preliminary experiments, resulting in a value of 0.1 for all our experiments. However, regarding μ and ϵ, we are reporting the results for different values to present a more complete view of our attack.

Algorithm 3 CatBack: Universal Backdoor for Tabular Data

Input: $D_{\text{orig}}, t, \mathscr{F}, \mu, \beta, \lambda, \epsilon$
Output: Backdoored model $\mathscr{F}'$
$D \leftarrow \text{Conv}(D_{\text{orig}})$ ▷ frequency-based encoding & tie-resolution
Train $\mathscr{F}$ on D
$D_{\text{nt}} \leftarrow \{(x, y) \in D \mid y \neq t\}$
for all $(x_i, y_i) \in D_{\text{nt}}$ **do**
    $s_i \leftarrow f_t(x_i)$ ▷ softmax score for class t
end for
Sort D_{nt} by s_i descending
$D_{\text{picked}} \leftarrow$ top $\mu |D_{\text{nt}}|$ samples
Initialize $\delta \leftarrow \mathbf{0} \in \mathbb{R}^d$
repeat
    for all $(x_i, y_i) \in D_{\text{picked}}$ **do**
        $\hat{x}_i \leftarrow \text{clip}(x_i + \delta)$
        $\ell_i \leftarrow -\log f_t(\hat{x}_i) + \beta \|\hat{x}_i - \text{Mode}(D)\|_1 + \lambda \|\hat{x}_i - \text{Mode}(D)\|_2^2$
    end for
    $\delta \leftarrow \delta - \eta \nabla_\delta \left(\frac{1}{|D_{\text{picked}}|} \sum_i \ell_i \right)$
    Round categorical dimensions of δ to nearest valid r'_{jl}
until convergence
Randomly select $D_{\text{sel}} \subset D$, $|D_{\text{sel}}| = \epsilon N$
for all $x_i \in D_{\text{sel}}$ **do**
    $\hat{x}_i \leftarrow \text{clip}(x_i + \delta), \quad \hat{y}_i \leftarrow t$
end for
$D' \leftarrow \text{Revert}\big((D \setminus D_{\text{sel}}) \cup \{(\hat{x}_i, \hat{y}_i)\}\big)$
Preprocess D' (OHE/embeddings/ordinal), train $\mathscr{F}'$ on D' **return** $\mathscr{F}'$

7.5.10. BLACK BOX ATTACK

For the black box scenario, we craft the perturbation using one of the white box models available to the attacker, implant the trigger in the samples, and release the poisoned dataset so that the black box model can be trained on it by a third-party victim. By this approach, we want to verify that our trigger is not tied to a specific model and can transfer to different architectures, making the attack more general.

7.6. EVALUATION AND RESULTS

We used five datasets (see Table 7.4) including Forest Cover Type (CovType)[5], Higgs Boson (HIGGS)[6], Bank Marketing[7], Credit Card Fraud Detection[8], and Adult Census Income (ACI)[9]. As previously mentioned, we picked four models used in previous studies showing the best performance [5, 18, 19], including XGBoost, TabNet, FTT, and Saint.

7.6.1. ENVIRONMENT AND SYSTEM SPECIFICATION

All experiments are conducted on a Red Hat Enterprise Linux 9 system equipped with Intel Xeon Platinum 8360Y and AMD EPYC 9334 CPUs, 1TiB RAM, and an NVIDIA H100

[5] https://archive.ics.uci.edu/dataset/31/covertype
[6] https://archive.ics.uci.edu/dataset/280/higgs
[7] https://www.kaggle.com/datasets/ruthgn/bank-marketing-data-set
[8] https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
[9] https://shap.readthedocs.io/en/latest/generated/shap.datasets.adult.html

and A100. Python 3.11.3 and Pytorch 2.5.1 were used for all experiments.

Table 7.4: Overview of the datasets used in experiments (after preprocessing).

	ACI	BM	CovType	Credit Card	HIGGS
Samples	32561	11162	588892	284807	11000000
Numerical features	5	6	10	30	28
Categorical features	7	9	44	0	0
Num. of classes	2	2	7	2	2
Class Ratio	(76%, 24%)	(53%, 47%)	(36%, 49%, 6%, 0.6%, 1.7%, 3%, 3.7%)	(99.8%, 0.2%)	(47%, 53%)

7.6.2. Evaluation Metrics

To assess the effectiveness and stealthiness of backdoor attacks, we utilize the following metrics:

1. **Attack Success Rate (ASR):** This metric quantifies the proportion of triggered inputs that the model classifies as the target label y_t:

$$ASR = \frac{1}{m} \sum_{j=1}^{m} \mathbb{I}\left(\mathscr{F}_{\theta^*}(\hat{\mathbf{x}}_j) = y_t\right),$$

   where $\mathscr{F}_{\theta^*}$ denotes the backdoored model, $\mathbb{I}$ is the indicator function, and m represents the number of poisoned inputs. A higher ASR signifies a more effective backdoor attack.
2. **Clean Data Accuracy (CDA):** This metric measures the model's accuracy on clean (unpoisoned) inputs and is defined as:

$$CDA = \frac{1}{n-m} \sum_{i=1}^{n-m} \mathbb{I}\left(\mathscr{F}_{\theta^*}(\mathbf{x}_i) = y_i\right),$$

   where n is the total number of inputs, and $n-m$ represents the count of clean inputs. CDA is compared against the Benign Accuracy (BA) of a clean model to determine whether the backdoor compromises the model's performance on legitimate data. A high CDA indicates minimal impact on the model's clean data performance, ensuring its stealthiness [26].

7.6.3. Benign Accuracies

We report the BA values in Tables 7.5 and 7.6. Table 7.5 demonstrates the BA values for pure numeric datasets HIGGS and Credit Card, and Table 7.6 includes the results for datasets containing both categorical and numerical features. All results are aligned with the accuracy values reported in previous studies and benchmarks [29, 19, 18].

7.6.4. Attack Results

For our primary experiments, we perform CatBack with $\mu = 1$, $\lambda = 0.1$, $\beta = 0.1$, and $\epsilon = [0.01, 0.02, 0.05, 0.1]$ with all the target classes possible within a chosen dataset.[10]

For the black box scenario, we choose the XGBoost as our target model. XGBoost is a decision tree model. The attacker cannot optimize the gradient-based perturbation in

[10] Full experiments results, including the effect of choosing different μ values, are provided in Section 7.6.5.

Table 7.5: BA results (%) for datasets containing only numerical columns.

	Credit Card	HIGGS
TabNet	99.87	77.85
Saint	99.94	79.89
FTT	99.95	79.44
XGBoost	99.95	76.23

Table 7.6: BA results (%) for different encoding methods

Dataset	Model	Method		
		OHE	Ordinal	**Ours**
ACI	XGBoost	85.905	85.183	85.398
	TabNet	85.015	85.521	85.460
	FTT	86.396	86.396	86.089
	Saint	86.366	86.366	86.320
BM	XGBoost	85.804	85.266	85.535
	TabNet	84.326	81.818	82.983
	FTT	83.565	87.013	82.176
	Saint	84.595	85.446	84.998
CovType	XGBoost	96.499	96.510	96.510
	TabNet	94.875	94.563	94.482
	FTT	96.176	96.302	96.116
	Saint	96.854	96.947	96.753

CatBack using XGBoost. Thus, it can be a good candidate for a black box attack. First, we craft our universal perturbation using another model to which we have white box access. In our experiments, we chose TabNet and optimized the perturbation so that it could successfully change TabNet's prediction to the target class. Once we craft our perturbation, we poison the training set by applying it to chosen samples and then train the XGBoost on the poisoned dataset. Finally, we measure CatBack's performance on the backdoored model using the same perturbation. Table 7.7 shows the attack performance with the selected hyperparameters and $\mu = 1$. For all results, the CDA values remain similar to their BA and are thus unaffected by CatBack. This is highly in favor of the attacker remaining stealthy with respect to the clean accuracy drop. With just 1% poisoning rate, CatBack can achieve more than 90% ASR in most cases. The results, though, depend on the dataset and target label. The μ hyperparameter extensively affects the trigger optimization overhead. This might not be noticeable for regular datasets where it takes seconds for the trigger to be obtained, but for large datasets like HIGGS (where, in our worst-case report, just the regular clean training of Saint might take hours), it is highly evident. As we show in Figures 7.2 to 7.6, μ also affects ASR for datasets with fewer samples (e.g., BM in Figure 7.3). With higher values for μ, more samples distant to the target class manifold get involved in the perturbation optimization process, thus making it more general and effective. This makes it easier for the perturbation to shift the distribution toward the target class.

On the other hand, bigger μ increases the chance of backdoor detection, as we see

later. The only exception we detected in experiments was the results for TabNet on the Credit Card dataset on target label 1 (see Figure 7.5). Although this did not happen with other models, we assume such results come from the dataset being heavily unbalanced (only around 0.1% of samples in the whole dataset belong to class number 1), making it difficult for a universal perturbation to perform perfectly on all random samples.

Another interesting observation is the results of a black box attack. We can notice that CatBack performs the same or even better in most cases on XGBoost than other models like TabNet. These observations suggest that CatBack can also be a very powerful threat in scenarios in which the poisoned dataset is provided to a third party (e.g., outsourced training), even though the attacker is unaware of the target model.

7.6.5. Detailed Attack Results

In this section, we have included the results from our experiments for all the hyperparameters we evaluated. In particular, in Figures 7.2 to 7.6, we show ASR (solid lines) and CDA (dotted lines) for our experiments for all datasets, models, target labels, and μ values. In Figure 7.2, we see that our attack achieves a performance of more than 80% in most cases. Additionally, the performance slightly increases as we increase the poisoning rate. In this case, both the μ and the target label do not significantly influence CatBack's performance. In Figure 7.3, however, the target label can affect the attack performance significantly in some cases. As BM is a relatively small dataset, we believe that the small imbalance of the two classes could lead to such differences. Class 0 corresponds to 53% of the dataset, and for this reason, the attack is slightly more effective when the target class is 0. Additionally, in Figure 7.3, when the FTT is used, we see that μ also affects CatBack's performance for both target labels, with $\mu = 1$ leading to the best performance due to bigger perturbation values. Similar results were seen for the rest of the datasets as shown in Figures 7.4 to 7.6.

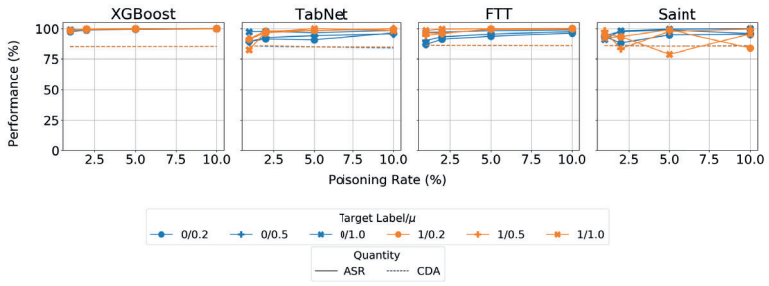


Figure 7.2: CatBack's ASR and CDA vs. the poisoning rate for different μ using the ACI dataset.

7.6.6. Baselines

We have also compared our attack with two baseline attacks. First, we tested Badnets [20]. To transfer Badnets to the tabular data, we applied the equivalent of an 8×8 square trigger in an MNIST image on each poisoned sample. In particular, the perturbation is 0.08% of all features (MNIST images are 28×28 pixels so $(8 \times 8)/(28 \times 28) = 0.08$). We chose a random value within the valid range for each feature for the trigger. We also ran experiments with

Table 7.7: CDA and ASR (CDA/ASR) for $\mu = 1$.

Dataset	Model	Target Label	ϵ			
			0.01	0.02	0.05	0.1
ACI	FTT	0	86.46/96.85	86.09/97.10	86.20/98.17	85.94/98.96
		1	86.34/98.50	86.35/99.62	85.94/99.77	86.10/100.0
	Saint	0	85.91/90.99	86.20/98.11	85.55/99.55	85.95/99.91
		1	85.95/92.35	85.74/93.38	85.51/78.84	85.52/95.55
	TabNet	0	85.67/97.39	85.40/97.62	84.83/96.53	84.25/98.42
		1	85.37/82.60	86.04/97.11	85.06/100.0	84.78/98.27
	XGBoost	0	85.09/99.11	85.26/99.05	85.24/99.74	85.32/99.91
		1	85.41/98.50	85.20/99.51	85.28/99.91	85.44/99.94
BM	FTT	0	83.70/87.33	86.52/95.75	86.25/99.96	86.30/100.0
		1	86.61/98.52	86.43/99.91	86.57/100.0	85.49/100.0
	Saint	0	85.09/96.33	84.51/98.88	84.68/100.0	85.85/99.96
		1	85.36/10.00	85.85/100.0	85.31/100.0	84.51/100.0
	TabNet	0	83.88/91.27	84.06/89.57	83.48/96.78	83.16/98.30
		1	83.74/84.73	82.22/88.22	82.80/97.49	79.94/96.95
	XGBoost	0	84.68/93.55	84.51/95.39	84.73/97.85	84.55/99.46
		1	84.86/95.21	84.51/97.27	84.95/99.24	84.55/99.96
CovType	FTT	0	95.76/99.98	96.23/100.0	96.21/100.0	95.90/100.0
		1	96.30/100.0	96.28/100.0	96.23/100.0	96.17/100.0
		2	96.31/99.99	96.15/100.0	96.29/100.0	96.03/100.0
		3	96.27/99.99	96.23/99.99	96.21/100.0	96.15/100.0
		4	96.16/99.98	96.20/100.0	96.15/100.0	96.03/100.0
		5	96.30/100.0	96.20/100.0	96.01/100.0	96.15/100.0
		6	97.12/99.99	96.86/100.0	97.09/100.0	97.01/100.0
	Saint	0	96.76/100.0	96.88/99.96	96.83/100.0	96.58/99.99
		1	96.81/99.98	96.73/99.97	96.79/99.70	96.65/100.0
		2	96.83/99.98	96.75/99.85	96.74/100.0	96.61/99.91
		3	96.83/99.97	96.84/99.84	96.77/99.85	96.49/99.87
		4	96.85/99.92	96.85/99.99	96.72/100.0	96.60/97.64
		5	96.93/99.97	96.94/100.0	96.89/99.98	96.54/99.95
		6	98.18/99.97	98.14/99.96	98.16/100.0	97.60/99.99
	TabNet	0	94.13/99.93	94.72/99.97	94.36/100.0	93.73/100.0
		1	94.10/99.90	94.54/99.99	94.37/100.0	94.38/100.0
		2	93.32/99.97	92.80/100.0	94.47/100.0	94.66/100.0
		3	94.53/99.95	91.07/100.0	94.21/100.0	94.22/99.98
		4	92.18/99.88	94.39/99.99	94.59/100.0	93.32/100.0
		5	93.20/99.97	94.75/99.91	94.70/100.0	94.75/100.0
		6	95.24/99.96	95.64/99.96	95.41/99.99	95.58/100.0
	XGBoost	0	96.51/99.98	96.49/100.0	96.44/100.0	96.32/100.0
		1	96.50/100.0	96.51/100.0	96.42/100.0	96.38/100.0
		2	96.55/99.99	96.49/100.0	96.42/100.0	96.38/100.0
		3	96.52/100.0	96.47/100.0	96.47/100.0	96.35/100.0
		4	96.51/99.99	96.47/99.99	96.46/100.0	96.40/100.0
		5	96.54/99.99	96.54/100.0	96.43/100.0	96.39/100.0
		6	98.73/99.99	98.69/100.0	98.71/100.0	98.56/100.0
Credit	FTT	0	99.95/99.86	99.95/99.86	99.92/99.86	99.94/99.87
		1	99.95/99.99	99.95/99.98	99.95/99.99	99.95/99.98
	Saint	0	99.95/99.87	99.96/99.85	99.96/99.86	99.96/99.86
		1	99.95/99.90	99.94/99.87	99.95/99.99	99.96/99.98
	TabNet	0	99.85/99.98	99.93/99.85	99.83/100.0	99.92/99.82
		1	99.93/99.71	96.93/95.15	99.93/99.95	99.91/86.18
	XGBoost	0	99.96/99.87	99.96/99.87	99.96/99.88	99.95/99.90
		1	99.96/99.93	99.96/99.97	99.96/99.95	99.96/99.99
Higgs	FTT	0	83.98/99.99	83.95/100.0	83.87/100.0	83.59/100.0
		1	83.89/100.0	83.96/100.0	83.95/100.0	83.81/100.0
	Saint	0	80.70/95.00	80.74/99.13	80.70/99.35	80.63/100.0
		1	80.67/94.94	80.64/99.95	80.59/95.58	80.48/99.95
	TabNet	0	77.70/99.98	78.47/100.0	78.18/100.0	78.49/100.0
		1	78.21/100.0	78.48/100.0	78.00/99.95	78.30/100.0
	XGBoost	0	77.38/100.0	77.35/100.0	77.36/100.0	77.35/100.0
		1	77.41/100.0	77.34/100.0	77.37/100.0	77.35/100.0

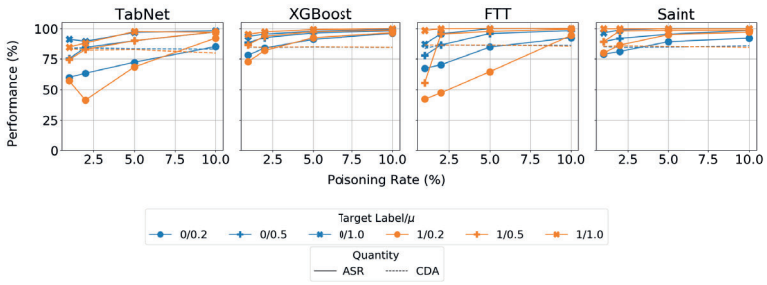


Figure 7.3: CatBack's ASR and CDA vs. the poisoning rate for different μ using the BM dataset.

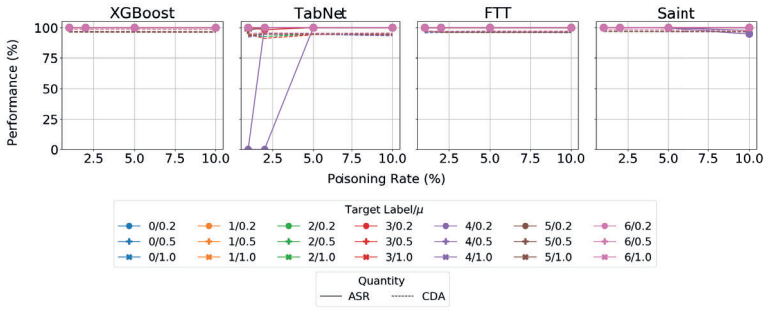


Figure 7.4: CatBack's ASR and CDA vs. the poisoning rate for different μ using the Forest Cover Type dataset.

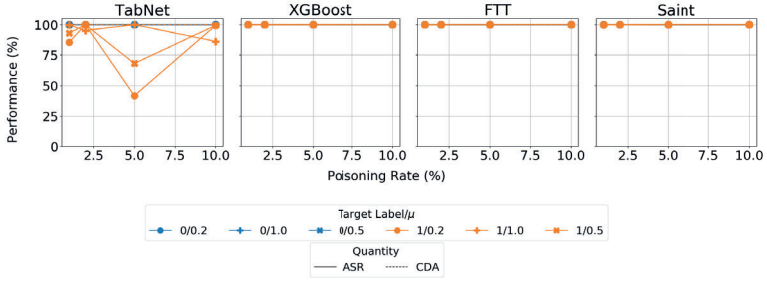


Figure 7.5: CatBack's ASR and CDA vs. the poisoning rate for different μ using the Credit Card dataset.

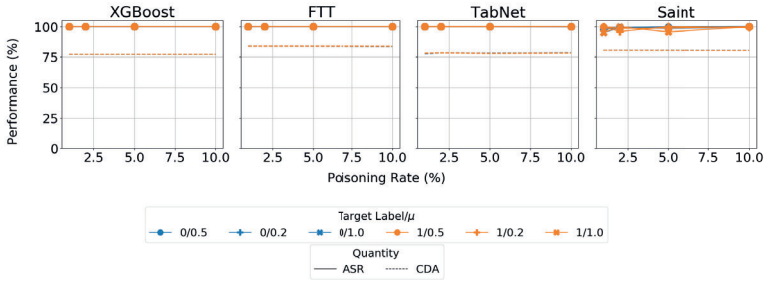


Figure 7.6: CatBack's ASR and CDA vs. the poisoning rate for different μ using the HIGGS dataset.

Tabdoor [36] using the in-bounds method. We did not try the out-of-bounds attack since the trigger uses values that are 10% higher than the maximum value of the poisoned features, which can be easily spotted as outliers. The same is true for the approach used in DBA, where the authors, to create a poisoned sample, changed the values in 6 features of the LOAN dataset using values larger than the maximum value for each of these features.

The results from these experiments are shown in Table 7.8. We did not include the CDA in the tables, as in all cases the performance of the poisoned models was similar to the clean model's performance. According to the tables, our attack performs similarly to the baselines for CovType, Credit Card, and Higgs. In these cases, we see that the attack performance is almost perfect (e.g., close to 100%) in all datasets, meaning that these settings are the easiest to attack. For this reason, CatBack cannot show significantly better performance than the baselines. On the other hand, CatBack performs significantly better for ACI (up to 95% and 44% better performance than Tabdoor and Badnets, respectively) and BM (up to 71% and 16% better performance for Tabdoor and Badnets) datasets. Additionally, our trigger can be applied to any data type (numerical and categorical), while the previous attacks were applied only to numerical data.

7.6.7. ADDITIONAL THREAT SCENARIO

As we mentioned in Section 7.5, the attacker reverts the poisoned dataset back to its original form after the poisoning is completed. In this way, the users can download and use the dataset in their own pipelines in its original form. $Revert(.)$ is an extra step of our attack, which could be omitted if our encoding method is used directly for the model training by the users, making our attack more efficient. In this case, our encoding method should not affect the model's performance.

To show that this is a viable scenario, we compared the performance of models trained using our encoding with the performance of models trained using ordinal or categorical encoding. From the results shown in Table 7.6, we see that our encoding method could be used as an alternative method for training models on tabular data without hurting the model's performance. This would make our attack easier, as in that case, the attacker should not revert the dataset back to its original form after the training completes. Additionally, our numerical encoding could facilitate input preprocessing in some cases (e.g., transformers like Saint and FTT) as it does not require an embedding

Table 7.8: Comparison of CatBack with baseline attacks. Catback Settings: $\mu = 1.0$, $\epsilon = 0.02$, target_label=1.

Dataset	Attack	FTT	SAINT	TabNet	XGBoost
ACI	Badnets	86.79	87.76	53.02	99.98
	Tabdoor	15.16	17.49	2.02	18.61
	CatBack	99.62	93.38	97.11	99.51
BM	Badnets	96.04	95.90	72.97	99.73
	Tabdoor	28.84	78.10	24.23	99.31
	CatBack	99.91	100.00	88.22	97.27
CovType	Badnets	100.00	100.00	100.00	100.00
	Tabdoor	99.96	99.98	99.77	100.00
	CatBack	100.00	99.97	99.99	100.00
Credit	Badnets	99.99	99.90	99.94	100.00
	Tabdoor	99.85	99.77	97.26	99.97
	CatBack	99.98	99.87	95.15	99.97
HIGGS	Badnets	100.00	100.00	99.98	100.00
	Tabdoor	100.00	93.63	99.99	100.00
	CatBack	100.00	99.95	100.00	100.00

layer as the model's input. Finally, our encoding does not result in an increase in the number of the table's columns, as this is common for methods like one-hot encoding, where each categorical column is converted to several binary columns.

7.6.8. EVALUATION ON VERTEX AI

Cloud computing has become the backbone of modern computing infrastructure, with widespread adoption across industries. Organizations of all sizes increasingly rely on cloud platforms to power machine learning workflows. Google's Vertex AI, Amazon SageMaker, and Azure Machine Learning are the top three services used by almost every company in the market [38]. Among them, Vertex AI provides a dedicated framework for tabular data through AutoML [16]. AutoML has an automated pipeline and algorithms to train and deploy the model. While specific algorithmic defenses within AutoML are not publicly detailed, they include automated data preprocessing (data cleaning, feature engineering, and outlier detection) and robust training practices (input validation, model armor, monitoring for prediction drift, etc.) [15].

As a real-world use case for our black box scenario, we decided to test whether CatBack can be effective on Google AutoML and see whether AutoML's internal mechanisms can mitigate or prevent our attack. Table 7.9 demonstrates the feasibility of such attacks in practice. In particular, in all cases, the attack is almost perfect (> 97%), and the platform did not provide any warnings about the data or the backdoor. Thus, even though this platform includes tools that perform automated data processing, the attack has not been spotted. Such an attack can affect the performance of deployed models in production through these platforms (highly likely for many instances, like the financial sector).

7.7. EVALUATION AGAINST DEFENSES

In this section, we investigate how Catback performs against benchmark defenses. We have evaluated our attack against various defense types (including backdoor detection

Table 7.9: CatBack Performance on Google AutoML. The attack settings: $\mu = 1.0$, $\epsilon = 0.02$, target_label=1.

Dataset	BA (%)	CDA (%)	ASR (%)
ACI	87.31	87.21	98.11
Credit Card	99.95	99.97	100.0
CovType	96.19	97.33	100.0
HIGGS	76.97	76.63	100.0
BM	86.18	87.86	96.99

and backdoor removal methods). We also evaluated the poisoned dataset on popular outlier detection methods.

7.7.1. SPECTRAL SIGNATURES

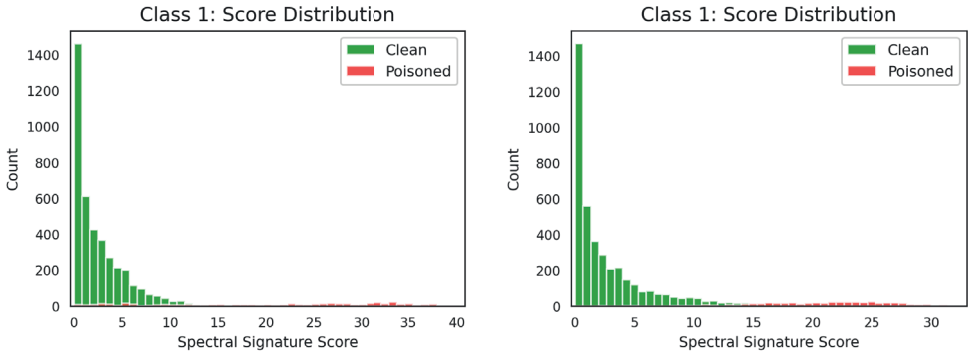
To evaluate CatBack's stealthiness, we implemented Spectral Signatures [42] on the poisoned datasets. Using the default settings from the paper, we assume the detector already knows ϵ and pinpoints $1.5 \times \epsilon$ of samples with the highest score as suspicious to remove them from the dataset. We experiment with different μ and ϵ values. As expected, there is an inverse relationship between ϵ and the stealthiness of the attack, as lower poisoning rates bypass the detection mechanism. Nonetheless, we observe that μ is a more important hyperparameter.

The higher μ values mean the universal perturbation is more general and should shift more random samples toward the decision manifold of the target class. On the other hand, smaller μ means choosing the samples closer to the decision manifold, which leads to a smaller perturbation needed for the shift. Thus, the poisoned samples are stealthier. For some datasets like BM, ASR is affected by μ value, especially when the poisoning rate is small (Figure 7.3). As the BM dataset is the smallest dataset in our experiments, we believe that there are fewer samples very close to the decision boundary, as it can be easier for the model to create a decision boundary in a sparse feature space. When we poison a dataset, we randomly select data from the training set, and by using a small poisoning rate in this case, we may select samples that are further away from the decision boundary. Thus, when μ is small and the trigger magnitude is also small, the trigger fails to alter the class of the poisoned data. However, if μ is large, the trigger magnitude is larger, which increases the ASR even if the samples are originally further away from the decision boundary. Thus, we expect it to leave a stronger effect on stealthiness as well.

We report the results for what be believe to be the least favorable condition for the attacker. Here, we chose the highest μ value of 1, $\epsilon = 0.02$ and target label 1. In this setting, the attack is mostly discoverable for the BM dataset (as we already expected since μ had the most influence on ASR for this dataset). Nonetheless, Spectral Signatures fails to remove or detect the poisoned samples in more than 75% cases. It is worth noting that with other settings, the defense failure rate is even higher, and for brevity, we only report the best defense performance. As an example, we see that by decreasing μ value to 0.5, it becomes more difficult to remove all poisoned samples for BM (Figure 7.7), and it entirely

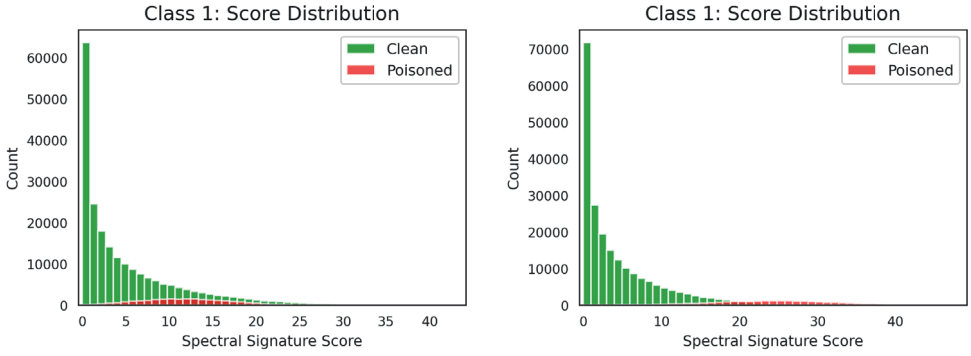
fails to detect poisoned samples for CovType with the FTT model (Figure 7.8).[11]

Moreover, in most cases, the score distributions for non-target classes are the same as the target class, which leads to removing the same number of samples from every non-target class, resulting in a performance reduction of the model on clean data. Previous works that used simple heuristics for the backdoor trigger, such as [36], resulted in more obvious triggers that could be spotted by Spectral Signatures. On the contrary, Spectral Signatures defense is not successful against CatBack, indicating that our attack is stealthy.



(a) $\mu = 0.5$. (b) $\mu = 1$.

Figure 7.7: Spectral Signatures: target class 1, Model: FTT, Dataset: BM, $\epsilon = 0.05$.



(a) $\mu = 0.5$. (b) $\mu = 1$.

Figure 7.8: Spectral Signatures: target class 1, Model: FTT, Dataset: CovType, $\epsilon = 0.05$.

7.7.2. Neural Cleanse

We evaluated our attack against Neural Cleanse [44] to measure how detectable the target class is. We adapted the defense for tabular data but adhered to the same hyperparameters and settings from the original paper. Our results show that Neural Cleanse fails to detect the target class. All the anomaly scores we got were below 2 (the threshold), which

[11] Due to space limitations, we include the rest of the figures in the supplementary file in our repository.

indicates a complete failure of the method. In fact, most of the scores show that the retrieved mask was around $0.67 \times MAD$, similar to clean scores. Interestingly, this is the standard deviation of the Laplace distribution used in calculating the z-score values for outlier detection, which refers to the normal range of values around the Median (for a z-score to be an outlier, the value should be 3 times more than that). In Table 7.10, we show our results for target_label=1. The results for other target labels are omitted as they demonstrate similar behavior.

Table 7.10: Anomaly Scores for Neural Cleanse. The attack settings: $\mu = 1.0$, $\epsilon = 0.02$, target_label=1.

Model	Dataset	Anomaly Score Backdoored/Clean
FTT	ACI	0.6745 / 0.6744
Saint	ACI	0.3699 / 0.6735
TabNet	ACI	0.6694 / 0.6731
FTT	BM	0.6745 / 0.6745
Saint	BM	0.6716 / 0.6730
TabNet	BM	0.6735 / 0.6744
FTT	CovType	0.0033 / 0.0091
Saint	CovType	1.5170 / 0.6742
TabNet	CovType	0.0046 / 1.9131
FTT	Credit Card	0.6745 / 0.6744
Saint	Credit Card	0.6744 / 0.6744
TabNet	Credit Card	0.6644 / 0.6856
FTT	HIGGS	0.6745 / 0.6844
Saint	HIGGS	0.6699 / 0.6838
TabNet	HIGGS	0.6740 / 0.6846

7.7.3. BEATRIX

Beatrix [34] leverages the second-order statistics of internal feature representations to identify anomalies introduced by backdoor triggers. The mechanism operates by computing Gram matrices from intermediate activations, which capture the correlations among features. Deviations from the expected behavior of clean samples are quantified using the median and the median absolute deviation (MAD). These deviations are further aggregated through a Kernel Maximum Mean Discrepancy (KMMD) metric and standardized to yield an anomaly score, denoted as J^*. High values of J^* (i.e., $ln(J^*) > 2$) indicate significant discrepancies from normal feature distributions, suggesting the potential presence of a backdoor. We evaluated our attack against Beatrix. The results indicate that Beatrix fails to detect the presence of Catback. For most cases, we got $ln(J^*) \approx -0.39$, which indicates the proximity of the poisoned Gram Matrix to the median. Table 7.11 showcases our results for target_label = 1.

Overall, the results from both Beatrix and Neural Cleanse highlight the high stealthiness of Catback, underscoring the importance of exercising caution when trusting third parties with trained models or ready-to-train datasets.

Table 7.11: Score results for Beatrix. The attack settings: $\mu = 1.0$, $\epsilon = 0.02$, target_label=1.

Model	Dataset	KMMD	$\ln(J^*)$
FTT	ACI	14.6131	-0.3938
Saint	ACI	74.0251	-0.3938
TabNet	ACI	43.0890	-0.3939
FTT	BM	19.5594	-0.3938
Saint	BM	28.0811	-0.3938
TabNet	BM	51.3612	-0.3938
FTT	CovType	3.7468	-0.3938
Saint	CovType	13.2158	-0.0416
TabNet	CovType	24.1161	0.1113
FTT	Credit Card	32.1099	-0.3938
Saint	Credit Card	0.0001	-9.2103
TabNet	Credit Card	39.4631	-0.3938
FTT	HIGGS	18.4841	-0.3938
Saint	HIGGS	24.2661	-0.3938
TabNet	HIGGS	101.8080	-0.3938

7.7.4. FINE PRUNING

To measure how robust our attack is, we evaluated it against Fine Pruning [31]. We pruned all feed-forward layers (FNN) within the transformers [17, 10] (SAINT and FTT) for up to 90% of neurons and fine-tuned the model for 5 epochs. For SAINT, we also pruned the penultimate layer to make the pruning stronger. Our results show that overall, fine pruning is not effective enough to prevent the attack from fully functioning. For SAINT, although it can restore the accuracy after fine tuning [27], ASR also remains considerably high. For FTT, however, Fine Pruning can successfully remove the attack for binary classification tasks with a smaller number of samples. Nevertheless, it fails to remove the backdoor if the dataset is large (e.g., Higgs). We hypothesize that this is because of a higher number of neurons in FNN being involved in learning the representations when facing more complicated datasets. With simpler datasets, few neurons can learn the binary manifold like a logistic regression task. This can also be observed when we observe the results for a multiclass dataset (Covtype), where the method fails in both preventing the attack and keeping the CDA high. To confirm this, we conducted extra experiments on another multiclass dataset: Poker Hand[12], consisting of 10 classes. First, we conduct the attack and evaluate its performance. Table 7.13 demonstrates the CatBack results on Poker Hand, with a successful high ASR. Then we perform the Fine Pruning whose results are shown in Table 7.14, and as we see, it likewise fails to mitigate the attack thoroughly.

7.7.5. EVALUATION WITH THE OUTLIER DETECTION ALGORITHMS

We have also investigated whether the poisoned samples created by CatBack can be detected using outlier detection methods. For this, we chose the two most popular outlier

[12] https://www.kaggle.com/datasets/hosseinah1/poker-game-dataset/data

Table 7.12: Fine Pruning Results for the pruning_rate = 0.9. The attack settings: $\mu = 1.0$, $\epsilon = 0.02$, target_label=1.

Model	Dataset	FP-CDA (%)	FP-ASR (%)
Saint	ACI	87.90	84.34
FTT	ACI	85.28	25.43
Saint	BM	86.94	99.96
FTT	BM	80.92	23.29
Saint	CovType	90.55	65.50
FTT	CovType	79.34	98.74
FTT	Credit Card	99.95	0.15
Saint	Credit Card	99.95	91.49
FTT	HIGGS	77.24	63.99
Saint	HIGGS	78.89	77.67

Table 7.13: Attack Results on Poker Dataset. The attack settings: $\mu = 1.0$, $\epsilon = 0.02$, target_label=1.

Model	BA (%)	CDA (%)	ASR (%)
Saint	99.78	99.99	100
FTT	99.79	99.96	99.90
TabNet	99.14	94.62	97.51
XGBoost	96.82	99.63	100

detection algorithms [21]: DBSCAN and Isolation Forest, which are commonly used in real-world applications, like fraud detection in finance or network intrusion detection [28]. We applied these two methods to our poisoned datasets, and both failed to detect and separate the poisoned and clean samples correctly. As a showcase, we report two of our results for the ACI (binary) and CovType (multiclass) datasets. Table 7.15 demonstrates the confusion matrices for Isolation Forest and DBSCAN, respectively. As we can observe, DBSCAN fails to achieve an F1-score of higher than 0.02 while Isolation Forest performs even worse by reaching a precision and recall of 0, meaning not a single poisoned sample can be detected. We omit the results for the remaining datasets as they showed a similar behavior.

7.8. ABLATION STUDIES

7.8.1. PARTIAL ACCESS TO TRAINING DATA

In our main threat model, we assume the attacker has full access to the complete training dataset. This might help the attacker observe more samples to calculate the trigger. To evaluate CatBack's effectiveness in stricter conditions, we assume a restricted threat model in which the attacker has a partial auxiliary dataset. We left only 10% of the dataset in the hands of the attacker and repeated the experiments. The results in Table 7.16 show that for datasets with a sufficient number of samples, the ASR remains almost the same, indicating that the trigger is still being calculated successfully. For smaller datasets like ACI and BM, with fewer samples, there is a noticeable drop in ASR; yet, in most cases, the

Table 7.14: Fine Pruning Results on Poker Dataset. The attack settings: $\mu = 1.0$, $\epsilon = 0.02$, target_label=1.

Model	FP-CDA (%)	FP-ASR (%)
Saint	99.96	66.41
FTT	67.55	74.91

Table 7.15: Confusion matrices for poisoned-sample detection (rows: Actual, columns: Predicted; *N*=Normal, *P*=Poisoned). Parameters: $\mu = 1.0$, $\epsilon = 0.02$, *target_label* = 1.

Isolation Forest						**DBSCAN**					
ACI			**Covtype**			**ACI**			**Covtype**		
	Pred.			*Pred.*			*Pred.*			*Pred.*	
Act.	N	P	*Act.*	N	P	*Act.*	N	P	*Act.*	N	P
N	25008	520	N	446217	9296	N	11324	14204	N	392434	63079
P	520	0	P	9296	0	P	426	94	P	7173	2123
$F_1 = 0$			$F_1 = 0$			$F_1 \approx 0.0127$			$F_1 \approx 0.0143$		

attack remains effective.

7.8.2. FREQUENCY MAPPINGS

Our Conv(·) mapping is strictly frequency-based. Each category is assigned a real value in $[0, 1]$ according to how often it occurs in the dataset (plus a small Δr to break ties). Thus, there is always a one-to-one mapping between frequency-mapped categories and their corresponding values in one-hot encoding or any other type of representation. This assures that our method preserves the same semantics needed by neural networks to learn and distinguish between categories of each feature. However, our method might induce an extra semantic to the domain knowledge for neural networks by representing the occurrence of each category when the network learns its relations with other co-occurred values in other features.

Another important aspect to notice is that when applying the trigger, the frequency of the categories might change from the original encoding. However, in our attack, the attacker reverts back the dataset to its original values, so the defender would not get suspicious of inconsistency between the frequency mapping values and the frequency of categories in the contaminated dataset, except by accessing the lookup table. Here, the

Table 7.16: ASR when the attacker controls only 10% of the data. The attack settings: $\mu = 1.0$, $\epsilon = 0.02$, target_label=1.

	Dataset				
Model	ACI	BM	CovType	Credit Card	HIGGS
FTT	90.63	60.55	100	99.42	100
Saint	93.35	81.33	100	99.65	100
TabNet	88.15	52.44	99.99	99.39	100
XGBoost	95.35	82.69	99.99	99.87	99.51

lookup table serves as a secret key, owned and used only by the attacker. One more interesting question is what happens if the user wants to use the raw frequency mappings as the default encoding. Then, the user would still not be able to observe this inconsistency, as the attacker conversion is only for poisoning purposes. After poisoning is done, the poisoned dataset can be converted again to another frequency mapping representation, and the new lookup table is given to the user, but the user never knows there was another lookup table owned by the attacker for poisoning conversion. Even in the worst-case scenario, if the user can obtain the lookup table, the frequency mappings of the clean and poisoned datasets are close, barely raising suspicion in the user. As an example, we demonstrate the Race and Relationship frequency mappings in the ACI dataset in Figure 7.9.



Figure 7.9: Race and Relationship values before and after poisoning in the ACI dataset.

7

7.8.3. PERFORMANCE UNDER DISTRIBUTION SHIFTS

To conduct further analysis, we designed two experiments for observing attack performance under distribution shifts:

1. In our first experiment, we decided to randomly change the distribution of the entire test set. For doing this, we iterate through all samples, and for each sample, we randomly pick 25% of all features, whether categorical or numerical (thus, for each sample, the chosen features are different). Then, for each chosen feature, we draw a new value using a Gaussian distribution (within the valid range) and substitute the new value with the old one. Then, we test the model on this newly sampled dataset (containing a new distribution). Table 7.17 shows that the attack is successful under the presumed condition.

2. In our second experiment, we implemented concept drift with retraining. Since there should be a pattern in concept drift for the model to learn, we first choose 25% of the columns randomly from the whole dataset (unlike the previous experiment, this time the chosen columns are fixed for all samples). Then, for each of these columns, if they are categorical, we simply replace them with the least common value for that category in the entire dataset. If the column is numerical, we perform a covariate shift on the values of that column.

Table 7.17: Distribution Shift Results on the ACI and Covtype datasets. Attack settings: $\mu = 1.0$, $\epsilon = 0.02$, target_label=1.

Dataset	Model	CDA (%)	ASR (%)
ACI	Saint	69.80	96.26
	Tabnet	68.75	90.78
Covtype	Saint	49.70	91.61
	Tabnet	63.07	98.68

To simulate covariate shift, we apply a sudden distributional change to a subset of samples in selected feature columns. Let $\mathscr{D}_{\text{train}}$ and $\mathscr{D}_{\text{test}}$ denote the training and test datasets respectively, and let $X \in \mathbb{R}^{n \times d}$ represent the feature matrix with n samples and d features.

Let $j \in \{1, \ldots, d\}$ denote the index of a selected numerical feature. We denote the values of this feature in the training set as:

$$X_{\text{train}}^{(j)} = \left\{ x_i^{(j)} \right\}_{i=1}^{n_{\text{train}}}.$$

We compute the empirical mean and standard deviation of the feature in the training set:

$$\mu_{\text{train}}^{(j)} = \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} x_i^{(j)}, \quad \sigma_{\text{train}}^{(j)} = \sqrt{\frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} \left(x_i^{(j)} - \mu_{\text{train}}^{(j)} \right)^2}.$$

We then define transformation parameters for drift:

* A random mean shift factor $\alpha \sim \mathscr{U}(0.1, 0.3)$

* A random scale factor $\beta \sim \mathscr{U}(0.1, 0.3)$

Using these, we compute:

$$\Delta\mu = \alpha \cdot \sigma_{\text{train}}^{(j)}, \quad \gamma = 1 + \beta.$$

The transformation applied to each selected test sample $x^{(j)}$ is:

$$x_{\text{drifted}}^{(j)} = \left(x^{(j)} + \Delta\mu \right) \cdot \gamma.$$

This results in a shifted and scaled version of the original feature, effectively altering both the first and second moments of the feature distribution. The operation introduces a sudden covariate shift.

We apply the concept drift on the entire test set and 25% of the train samples randomly. Then we perform a complete retraining of the model on the drifted training set and evaluate the attack on the test set. The results are demonstrated in Table 7.18. Results suggest that concept drift and retraining together can impact the ASR; however, this highly depends on the model.

As a takeaway, we can regard this as a possible solution to mitigate the attack efficacy. Thus, by monitoring and responding to distribution shifts (scheduled retaining), practitioners can remove implanted triggers over time.

Table 7.18: Concept Drift (with retraining) Results on the ACI and Covtype datasets. Attack settings: $\mu = 1.0$, $\epsilon = 0.02$, target_label=1.

Dataset	Model	CDA (%)	ASR (%)
ACI	Saint	84.08	99.43
	Tabnet	85.23	19.38
Covtype	Saint	92.32	91.5
	Tabnet	92.24	24.29

7.9. LIMITATIONS AND DISCUSSION

Imperceptibility refers to how unnoticeable the trigger is to a human observer or simple statistical checks. Its focus is on the input space (features), and the goal is to make the trigger visually or statistically subtle so it does not raise suspicion [37]. Stealthiness, however, usually refers to how difficult the attack is to detect by automated defenses or manual inspections, not just in the input, but also in model behavior, the training process, or data patterns. Its focus is on the training process, model internals, and activation patterns [32]. Its goal is to make the attack difficult for defenders to detect using anomaly detection, activation clustering, or mitigating techniques. Thus, the attack could be perceptible, yet highly stealthy, hiding from all detection methods [32].

A variety of domain-specific metrics have been proposed to quantify imperceptibility in non-tabular settings:

- **Image Domain:** Metrics based on L_p norms (e.g., L_2, L_∞) measure the magnitude of pixel perturbations, while perceptual similarity measures such as Structural Similarity Index (SSIM) and Learned Perceptual Image Patch Similarity (LPIPS) model human visual perception to assess perturbation visibility [46, 50, 45]. Frequency-domain metrics (e.g., DCT distortion) further ensure that the trigger does not introduce atypical spectral components.

- **Text Domain:** Metrics such as word-embedding distance and perplexity change quantify the semantic drift introduced by triggers. Approaches like BLEU and ROUGE scores, alongside language model-based fluency measures, evaluate whether inserted or substituted tokens remain coherent in the sentence context.

- **Voice Domain:** Imperceptibility is measured by signal-to-noise ratio (SNR), Mel-cepstral distortion (MCD), and perceptual evaluation of speech quality (PESQ) to ensure that the backdoor trigger (e.g., audio watermark) remains inaudible or indistinguishable from natural background noise.

One of the limitations we face during the current study is the lack of an imperceptibility metric in the tabular domain. Although we design our backdoor to be stealthy, there is no accepted measure in the community so that we can calculate how imperceptible the backdoor trigger is in tabular entry. Though designing and proposing a new metric

specific to tabular data is out of the scope of this work, we try to illustrate some aspects of a good imperceptibility metric tailored for tabular data, and hope this can provide a good insight for future studies. Similar to BlEU or SSIM, we assume that there is a reference sample for the given synthetic sample (here, we consider the original clean sample as the reference and its corresponding backdoored sample as the synthetic). Let us denote the synthetic row as

$$\mathbf{x} = (x_1, x_2, \ldots, x_F),$$

against a real reference row:

$$\mathbf{y} = (y_1, y_2, \ldots, y_F).$$

We assume that there is a tabular imperceptibility score denoted as TIS. We believe that TIS should be a function that is dependent on two important factors:

$$TIS = f(FLS, IF).$$

- Feature-Level Similarity (FLS): A per-feature score that measures how similar each candidate feature is to its reference counterpart.
- Interaction Fidelity (IF): A penalty/bonus component that captures the similarity of pairwise (or higher-order) interactions among features.

We conjecture that a good TIS should have these values calculated in its process, either explicitly or implicitly. FLS calculation might be just a distance measure (e.g., L_2 distance). However, calculating IF might be more complicated and vary from dataset to dataset. For example, we assume there is a tabular dataset consisting of different columns of patient data. Age and BP (bacterial peritonitis) columns might be highly correlated with each other. However, this is something that could be known only by specialists or researchers who obtained the correlation function between these features (e.g., through linear regression). Thus, formulating these correlation functions varies significantly between each dataset and may require the assistance of a specialist in that field. We leave this as an open question that might be an interesting topic for future studies.

7.10. RELATED WORK

The first backdoor attacks were introduced in [20, 9]. In these works, the authors demonstrated that an attacker can insert a backdoor into a trained model just by altering a few training samples. A large number of different attacks have been introduced since then, most of them targeting computer vision [29], but also targeting different domains like graph neural networks [47], language models [8], or speech processing [49].

Only a few works studied backdoors on tabular data. [48] performed distributed backdoor attacks on federated learning. For a backdoor on tabular data, the authors performed a feature importance ranking and chose the 6 least important features. For each of those chosen features, they set their value to an outbound number larger than the maximum value in the dataset. [36] conducted a more comprehensive study on backdoors in tabular data, resulting in the Tabdoor attack. Contrary to [48], the authors observed less of a relation between feature importance and attack success rate. Since the authors

investigated more datasets, they suggested that using the most important features results in a slightly higher attack success rate. To improve over [48], they proposed two different attacks: Out-of-bounds and In-bounds attacks. For Out-of-bounds, they followed the same idea as in [48], but with fewer features. The authors tried 1, 2, and 3 features and set their values to 10% higher than the maximum values of each feature in the whole dataset. Since the Out-of-bounds attacks can be easily discovered as outliers, [36] performed the In-bounds attack as their main contribution. For In-bounds, they chose the top 3 most important features within the dataset and set their values to the most common value for that feature. As presented in previous sections, CatBack works significantly better than Tabdoor, especially for more complex datasets.

[33] performs a data-free backdoor attack. They used an approach similar to teacher-student type of training to induce the trigger pattern in the victim model without accessing the dataset. For this, they collect a substitute dataset (which may be irrelevant to the main dataset) and try to implant the trigger value inside the substitute dataset. For tabular data, they chose ACI as their main task dataset and CovType as the substitute. They selected two features of ACI (fnlwgt, sex) with a fixed value and implanted them in the Covtype (with 20% poisoning rate), while changing the label to target, and then fine-tuned the victim model. Unfortunately, due to a significantly different setup from ours (different models, types of attacks, and different poisoning rates), direct comparison with our work is not possible. [35] performs a backdoor on Binary click-through rate (CTR) datasets using a Factorization model (Deep-FM). To do so, they conducted a feature importance ranking based on the Deep-FM embedding layer (they consider features with higher average weight values to be more important). Then, they select the top 4 most important features and set them to the least frequent value. However, the authors did not provide the code, and we could not reproduce the results. For this reason, we did not consider this attack in our evaluation.

7.11. ETHICAL CONSIDERATIONS

Our work investigates backdoor attacks for tabular data, highlighting how efficient transformation of input data can lead to high-performing and stealthy backdoors. While our goal is to increase awareness about this attack vector, considering the prevalence of tabular data in real-world applications, we acknowledge that disclosing such vulnerabilities could potentially be exploited for malicious purposes. All experiments conducted in this research were designed to avoid exposing sensitive data or causing real-world harm. Evaluations were performed in controlled environments, using publicly available datasets. Despite these precautions, we acknowledge that revealing any new attack carries some risk. Still, we believe that transparent disclosure of such vulnerabilities, combined with responsible communication, benefits the community.

7.12. CONCLUSIONS AND FUTURE WORK

In this work, we have introduced a targeted backdoor attack on tabular data by crafting a universal trigger pattern that can be added to any input within the domain. By leveraging frequency-based floating-point mapping for categorical values, we unify all feature types into a continuous space and enable a single, elastic-net-regularized perturbation to

reliably shift any sample toward an adversarial target. By enabling the trigger to involve any column of a table, including categorical features, we broaden the attack surface.

We evaluated CatBack across five widely used benchmark datasets and four distinct models, achieving near-perfect attack success rates (up to 100%) with negligible impact on clean-data accuracy, under constraints such as 1% poisoning rate or only 10% auxiliary data access. Moreover, our evaluation against state-of-the-art defenses shows that CatBack remains largely undefeated, and it readily transfers to real-world cloud services, such as Google Vertex AI.

These findings underscore the need for future studies to rethink trust assumptions around tabular pipelines, including the development of custom defenses for tabular data, such as preprocessing-agnostic detection strategies, and incorporating robust certification for mixed-type inputs. Future work could extend CatBack to dynamic, sample-specific triggers, investigate defenses based on causal feature attribution, and evaluate these threats in federated or privacy-preserving settings. Ultimately, securing mission-critical domains such as finance and healthcare will require holistic frameworks that treat tabular ML with the same adversarial scrutiny long applied to vision and language.

REFERENCES

[1] Gorka Abad et al. “A Systematic Evaluation of Backdoor Trigger Characteristics in Image Classification”. In: *arXiv preprint arXiv:2302.01740* (2023).

[2] Amazon Web Services. *Built-in SageMaker AI Algorithms for Tabular Data*. https://docs.aws.amazon.com/sagemaker/latest/dg/algorithms-tabular.html. Amazon SageMaker Developer Guide, Accessed 22 April 2025.

[3] Sercan Ö Arik and Tomas Pfister. “Tabnet: Attentive interpretable tabular learning”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 35. 2021, pp. 6679–6687.

[4] Eugene Bagdasaryan and Vitaly Shmatikov. “Blind backdoors in deep learning models”. In: *30th USENIX Security Symposium (USENIX Security 21)*. 2021, pp. 1505–1521.

[5] Vadim Borisov et al. “Deep neural networks and tabular data: A survey”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[6] Francesco Cartella et al. “Adversarial attacks for tabular data: Application to fraud detection and imbalanced data”. In: *arXiv preprint arXiv:2101.08030* (2021).

[7] Tianqi Chen and Carlos Guestrin. “Xgboost: A scalable tree boosting system”. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[8] Xiaoyi Chen et al. “Badnl: Backdoor attacks against nlp models with semantic-preserving improvements”. In: *Proceedings of the 37th Annual Computer Security Applications Conference*. 2021, pp. 554–569.

[9] Xinyun Chen et al. “Targeted backdoor attacks on deep learning systems using data poisoning”. In: *arXiv preprint arXiv:1712.05526* (2017).

[10] Kevin Clark et al. “What does bert look at? an analysis of bert’s attention”. In: *arXiv preprint arXiv:1906.04341* (2019).

[11] Jillian M Clements et al. “Sequential deep learning for credit risk monitoring with tabular financial data”. In: *arXiv preprint arXiv:2012.15330* (2020).

[12] Łukasz Delong and Anna Kozak. “The use of autoencoders for training neural networks with mixed categorical and numerical features”. In: *ASTIN Bulletin: The Journal of the IAA* 53.2 (2023), pp. 213–232.

[13] Amit Dhurandhar et al. “Model Agnostic Contrastive Explanations for Classification Models”. In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* (2024).

[14] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256.

[15] Google Cloud. *Google Cloud AI and Security Documentation*. https://cloud.google.com/vertex-ai/docs/model-monitoring/monitor-explainable-ai, https://cloud.google.com/security-command-center/docs/model-armor-overview, https://cloud.google.com/security-command-center/docs/security-posture-overview. Accessed: 18 April 2025. 2025.

[16] Google Cloud. *Train models with Vertex AI (Tabular data)*. https://cloud.google.com/vertex-ai/docs/training-overview#tabular. Accessed: 23 April 2025. 2025.

[17] Mitchell A Gordon, Kevin Duh, and Nicholas Andrews. “Compressing bert: Studying the effects of weight pruning on transfer learning”. In: *arXiv preprint arXiv:2002.08307* (2020).

[18] Yury Gorishniy et al. “Revisiting deep learning models for tabular data”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 18932–18943.

[19] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. “Why do tree-based models still outperform deep learning on typical tabular data?” In: *Advances in neural information processing systems* 35 (2022), pp. 507–520.

[20] Tianyu Gu et al. “BadNets: Evaluating Backdooring Attacks on Deep Neural Networks”. In: *IEEE Access* (2019).

[21] Victoria Hodge and Jim Austin. “A survey of outlier detection methodologies”. In: *Artificial intelligence review* 22 (2004), pp. 85–126.

[22] Arthur E. Hoerl and Robert W. Kennard. “Ridge regression: Biased estimation for nonorthogonal problems”. In: *Technometrics* 12.1 (1970), pp. 55–67.

[23] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. “Handcrafted backdoors in deep neural networks”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 8068–8080.

[24] Byunggill Joe et al. “Exploiting Missing Value Patterns for a Backdoor Attack on Machine Learning Models of Electronic Health Records: Development and Validation Study”. In: *JMIR Medical Informatics* 10.8 (2022), e38440.

[25] Guolin Ke et al. “Lightgbm: A highly efficient gradient boosting decision tree”. In: *Advances in neural information processing systems* 30 (2017).

[26] Stefanos Koffas et al. “A Systematic Evaluation of Backdoor Attacks in Various Domains”. In: *Embedded Machine Learning for Cyber-Physical, IoT, and Edge Computing: Use Cases and Emerging Challenges*. Springer, 2023, pp. 519–552.

[27] François Lagunas et al. “Block pruning for faster transformers”. In: *arXiv preprint arXiv:2109.04838* (2021).

[28] Md Tahmid Rahman Laskar et al. “Extending isolation forest for anomaly detection in big data via K-means”. In: *ACM Transactions on Cyber-Physical Systems (TCPS)* 5.4 (2021), pp. 1–26.

[29] Yiming Li et al. *Backdoor Learning: A Survey*. 2022. arXiv: 2007.08745 [cs.CR].

[30] Bo Liu et al. “When machine learning meets privacy: A survey and outlook”. In: *ACM Computing Surveys (CSUR)* 54.2 (2021), pp. 1–36.

[31] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. “Fine-pruning: Defending against backdooring attacks on deep neural networks”. In: *Research in Attacks, Intrusions, and Defenses (RAID)*. Springer. 2018, pp. 273–294.

[32] Zeyan Liu et al. “Hide and Seek: on the Stealthiness of Attacks against Deep Learning Systems”. In: *European Symposium on Research in Computer Security*. Springer. 2022, pp. 343–363.

[33] Peizhuo Lv et al. “A data-free backdoor injection approach in neural networks”. In: *32nd USENIX Security Symposium (USENIX Security 23)*. 2023, pp. 2671–2688.

[34] Wanlun Ma et al. “The "Beatrix" Resurrections: Robust Backdoor Detection via Gram Matrices”. In: *30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*. The Internet Society, 2023. URL: https://www.ndss-symposium.org/ndss-paper/the-beatrix-resurrections-robust-backdoor-detection-via-gram-matrices/.

[35] Lingshuo Meng, Xueluan Gong, and Yanjiao Chen. “BAD-FM: Backdoor Attacks Against Factorization-Machine Based Neural Network for Tabular Data Prediction”. In: *Chinese Journal of Electronics* 33.4 (2024), pp. 1077–1092. DOI: 10.23919/cje.2023.00.041.

[36] Bart Pleiter et al. “Tabdoor: Backdoor Vulnerabilities in Transformer-based Neural Networks for Tabular Data”. In: *arXiv preprint arXiv:2311.07550* (2023).

[37] Yao Qin et al. “Imperceptible, robust, and targeted adversarial examples for automatic speech recognition”. In: *International conference on machine learning*. PMLR. 2019, pp. 5231–5240.

[38] Felix Richter. *Amazon and Microsoft Stay Ahead in Global Cloud Market*. Accessed: 24 April 2025. 2024. URL: https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/.

[39] Gowthami Somepalli et al. “Saint: Improved neural networks for tabular data via row attention and contrastive pre-training”. In: *arXiv preprint arXiv:2106.01342* (2021).

[40] Behrad Tajalli et al. “ELMs Under Siege: A Study on Backdoor Attacks on Extreme Learning Machines”. In: *Proceedings of the 2024 Workshop on Artificial Intelligence and Security*. 2024, pp. 125–136.

[41] Robert Tibshirani. “Regresion shrinkage and selection via the Lasso”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[42] Brandon Tran, Jerry Li, and Aleksander Madry. “Spectral signatures in backdoor attacks”. In: *Advances in neural information processing systems* 31 (2018).

[43] Dennis Ulmer, Lotta Meijerink, and Giovanni Cinà. “Trust issues: Uncertainty estimation does not enable reliable ood detection on medical tabular data”. In: *Machine Learning for Health*. PMLR. 2020, pp. 341–354.

[44] Bolun Wang et al. “Neural cleanse: Identifying and mitigating backdoor attacks in neural networks”. In: *S&P*. 2019.

[45] Hang Wang et al. “Mm-bd: Post-training detection of backdoor attacks with arbitrary backdoor pattern types using a maximum margin statistic”. In: *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2024, pp. 1994–2012.

[46] Zhou Wang et al. “Image Quality Assessment: From Error Visibility to Structural Similarity”. In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612.

[47] Zhaohan Xi et al. “Graph backdoor”. In: *30th {USENIX} Security Symposium ({USENIX} Security 21)*. 2021.

[48] Chulin Xie et al. “Dba: Distributed backdoor attacks against federated learning”. In: *ICLR*. 2019.

[49] Tongqing Zhai et al. “Backdoor attack against speaker verification”. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 2560–2564.

[50] Richard Zhang et al. “The unreasonable effectiveness of deep features as a perceptual metric”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 586–595.

[51] Hui Zou and Trevor Hastie. “Regularization and variable selection via the elastic net”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2 (2005), pp. 301–320.

8

A Systematic Evaluation of Backdoor Attacks in Various Domains

Deep learning found its place in various real-world applications, where many also have security requirements. Unfortunately, as these systems become more pervasive, understanding how they fail becomes more challenging. While there are multiple failure modes in machine learning, one category received significant attention in the last few years - backdoor attacks. Backdoor attacks aim to make a model misclassify some of its inputs to a preset-specific label while other classification results would behave normally.

While many works investigate various backdoor attacks and defenses for different domains, no works aim to provide a systematic comparison of backdoor attacks for different scenarios. This work considers backdoor attacks in image, sound, text, and graph domains and provides a comparative analysis of their respective strengths.

This chapter has been published as S. Koffas, B. Tajalli, J. Xu, M. Conti, S. Picek (2023). A Systematic Evaluation of Backdoor Attacks in Various Domains. *In Embedded Machine Learning for Cyber-Physical, IoT, and Edge Computing: Use Cases and Emerging Challenges (pp. 519-552).* Cham: Springer Nature Switzerland.

8.1. INTRODUCTION

In the last few years, deep learning has become very popular, and it has been applied to a variety of applications like computer vision [29], machine translation [54], speech recognition [18], and game playing [44]. It is also used in safety and security-critical applications like autonomous driving [12], malware detection [9], biometric-based user authentication [7], and side-channel analysis [40]. Such systems commonly need large datasets to train reliable models that generalize well and perform adequately with unseen data. However, large datasets are often scrapped from untrusted sources on the web [2, 11]. Additionally, the hardware needed to train such models can be very expensive and is not always available to developers who want to embed some machine learning functionality into their applications. Thus, a new programming paradigm has emerged: Machine Learning as a Service (MLaaS), made possible by the recent advances in cloud computing. These new trends lead to novel attack vectors that adversaries can exploit.

One of these attack vectors is the backdoor attack [19]. In this attack, an adversary embeds a secret functionality into a trained model, activated only if the model's input contains a specific property (trigger). At the same time, for any input that does not include the trigger, the model behaves as expected to avoid raising any suspicions. Most of the designed attacks in the literature target computer vision applications [31], but recently different applications have been targeted. In particular, backdoor attacks were shown in text classification [10, 6], audio recognition [62, 28], graph data [55, 57], federated learning [4, 45], and reinforcement learning [58]. A backdoor attack can be dangerous as machine learning is used in many security-related applications. In [19], the authors showed that a stop sign with a small post-it note could be identified as a speed limit by a compromised autonomous vehicle with serious consequences to its passengers and pedestrians. AI-enabled applications like spam-filtering [37], speaker identification [62], or malware detection [42] could also be bypassed if the model used contains a backdoor. Thus, backdoor attacks pose a serious threat, and it is required to understand the limits of such attacks to provide better defenses.

This work explores the effects of various trigger characteristics on the backdoor attack. In particular, we implement backdoor attacks with triggers of varying sizes, positions, and poisoning rates and apply them to four different domains (image, text, sound, and graph data). With it, we aim to better understand backdoor attacks and find common properties among different domains.

In [47], the authors claimed that the backdoor attack becomes ineffective when the adversary cannot alter the training labels and is forced to poison only samples from the target class. In this case, the model cannot learn a strong connection between the trigger and the target class as more substantial features from the target class are learned. This behavior is reasonable and well justified but only supported by one experiment with the CIFAR10 dataset. Here, we aim to test this claim in image classification but also in different domains, like text and sound classification.

Our contributions are:

- We run extensive experiments in different application domains (image, text, audio, and graph data) and systematically evaluate the effect of various trigger characteristics on the backdoor attack.

- We investigate two different attacks in each application and verify that the clean-label attack is not very effective as it may require large poisoning rates to achieve a high attack success rate. However, this attack could work by choosing more effective triggers without changing the poisoning rate.
- We show that in most cases, the backdoor's effectiveness increases as the trigger size increases.

8.2. BACKGROUND

8.2.1. COMPUTER VISION

Today, the computer vision domain covers diverse use cases and concepts within, ranging from capturing raw data to image pattern extraction and interpreting information from those images. It is mostly a combination of concepts, ideas, and techniques of pattern recognition, digital image processing, artificial intelligence (AI), and computer graphics [53]. Computer vision aims to provide the capability for a system to identify and perceive the visual world in the same way as human vision does. Recently, by applying AI techniques, including deep neural networks, the machines even outperformed humans in several tasks [13].

Nowadays, there are multiple applications of computer vision in our daily life, e.g., weather prediction, medical cases, sports and entertainment, industry and production lines, and human-computer interaction [25, 36, 24, 51, 60, 46, 49]. While the use cases and applications are becoming broader and more prevalent in our everyday lives, security issues regarding the techniques and algorithms are also becoming a significant challenge to deal with.

8.2.2. NATURAL LANGUAGE PROCESSING

Natural language processing (NLP) is at the intersection of computational linguistics, computer science, and artificial intelligence. It aims to make machines that understand human language and reason about it. NLP is an umbrella term that covers many different applications that deal with human language in both spoken and written formats. Applications that belong to natural language processing are, among others, speech recognition, speaker identification, question answering, text sentiment analysis, hate speech detection, natural language generation (speech-to-text and text-to-speech models), spam detection, and text translation. Initially, NLP was based on static rules, but now it uses deep learning for most tasks [23].

Recent advances in NLP have led to very efficient human-computer interfaces that have been broadly deployed. Virtual assistants like Siri and Google assistant and popular IoT devices like Amazon Alexa have been widely used with great success. However, such applications open up new attack vectors that put the user's security and privacy at risk. Therefore, before their wide adoption in the industry, we must ensure that such systems work securely.

8.2.3. GRAPH DATA

Many real-world applications can be modeled as graphs, such as social networks, gene interactions, and transport networks. Similar to the great success of deep learning models

in, e.g., image classification and natural language processing, deep graph models (graph neural networks - GNNs) have also achieved promising performance in processing graph data for different tasks, e.g., the graph classification task and node classification task.

Graph Neural Networks (GNNs). GNNs take a graph G as an input, including its structure information and node features, and learn a representation vector (embedding) for each node $v \in G$, z_v, or the entire graph, z_G. Modern GNNs follow a neighborhood aggregation strategy, where one iteratively updates the representation of a node by aggregating representations of its neighbors. After k iterations of aggregation, a node's representation captures both structure and feature information within its k-hop network neighborhood. Formally, the k-th layer of a GNN is (e.g., GCN [26], GraphSAGE [20], and GAT [48]):

$$Z^{(k)} = AGGREGATE(A, Z^{(k-1)}; \theta^{(k)}), \tag{8.1}$$

where $Z^{(k)}$ are the node embeddings in the matrix form computed after the k-th iteration and the $AGGREGATE$ function depends on the adjacency matrix A, the trainable parameters $\theta^{(k)}$, and the previous node embeddings $Z^{(k-1)}$. $Z^{(0)}$ is initialized as G's node features.

For the node classification task, the node representation $Z^{(k)}$ of the final iteration is used for prediction. For the graph classification task, the READOUT function pools the node embeddings from the final iteration K:

$$z_G = READOUT(Z^{(K)}). \tag{8.2}$$

READOUT can be a simple permutation invariant function such as summation or a more sophisticated graph-level pooling function [59, 63].

Graph-level classification. Graph-level classification aims to predict the class label(s) for an entire graph [63]. The end-to-end learning for this task can be realized using graph convolutional layers and readout layers. While graph convolutional layers are responsible for extracting high-level node representations, the readout layer collapses node representations of each graph into a graph representation. By applying a multilayer perceptron and a Softmax layer to graph representations, one can build an end-to-end framework for graph classification.

Node-level classification. Given a graph with a few labeled nodes, GNNs can learn a robust model that effectively identifies the class labels for the unlabeled nodes [26]. In a node-level classification task, there are two types of training settings - inductive and transductive. In an inductive setting, the unlabeled nodes are not seen during training, while in a transductive setting, the test nodes (but not their labels) are also observed during the training process. The transductive training setting is popular, and in this work, we used a backdoor attack in the transductive node-level classification task.

8.2.4. BACKDOOR ATTACKS

Backdoor attacks aim to make a model misclassify some of its inputs to a preset-specific label while other classification results behave normally. This misclassification is activated when a specific property is included in the model input. This property is called the trigger and can be anything the targeted model understands. For instance, a random pixel pattern [19, 7] or an actual item [52] in computer vision, a specific phrase in text

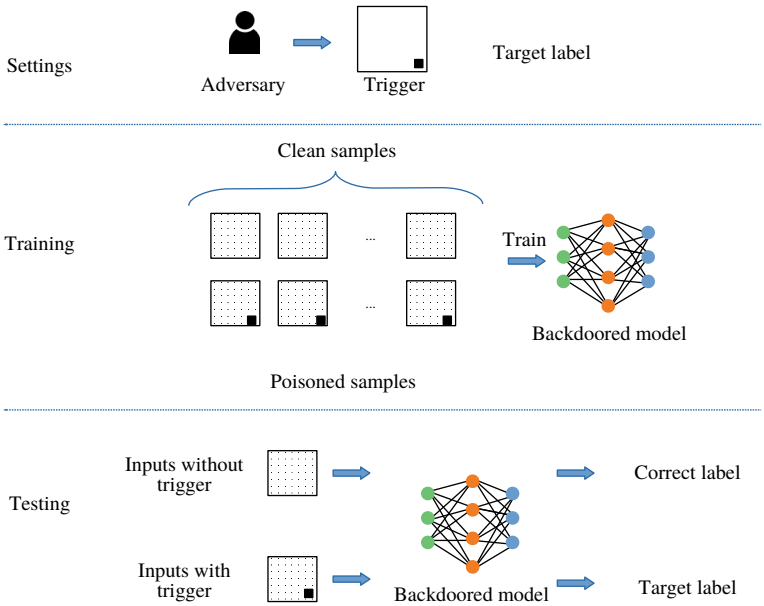


Figure 8.1: Framework for the backdoor attack.

classification [32], a tone in speech recognition [28], or a subgraph with specific properties in graph data [55]. The framework for the backdoor attack is shown in Figure 8.1.

The first backdoor attacks targeted computer vision [19, 7] under a simple threat model, where an adversary could inject a small portion of poisoned data into the training dataset. In particular, the adversary injects into the training dataset data stamped with a trigger that belongs to the target class. As a result, the trained model strongly associates this pattern with the target class, and whenever it is added to an input, the classification result will be the target class. Recent trends in machine learning like Machine Learning as a Service (MLaaS), outsourced training, transfer learning, and crowdsourced datasets have made this setup possible.

In MLaaS, a cloud provider provides a pay-per-request API[1] that can be used for predictions. However, the user can only use such an API as a black box without being able to verify how the model makes its predictions. Similarly, during outsourced training, the user's model is trained on the cloud and returned to the user after the training ends. Due to the lack of formal verification tools for the trained models, the user can never verify that the returned model does not contain any backdoors. Furthermore, in [19], the authors showed that a backdoor could remain effective even after a poisoned model was repurposed through transfer learning. Large crowdsourced datasets like ImageNet [11] and Mozilla's common voice [2] are so vast that cannot be exhaustively verified [39]. Thus, an adversary could inject a few poisoned samples resulting in the backdoored models.

This threat can pose real challenges as an adversary could bypass a face identification

[1] https://aws.amazon.com/transcribe/

biometric access control system [7] or force an autonomous vehicle to ignore a stop sign and continue its course [19]. For this reason, backdoor attacks became very popular among researchers resulting in many novel attacks and countermeasures [15]. Novel attacks are not only limited to data poisoning but can also be based on code poisoning [3] or the direct modification of the model's parameters [22]. At the same time, due to the inability to completely understand how a deep learning model works and the lack of formal verification methods about a model's functionality, most countermeasures are empirically based on specific assumptions [50, 5, 16]. Unfortunately, in most cases, an adaptive attacker with a slightly different approach could bypass such defenses [30, 43, 8].

There are several variations of the backdoor attack resulting from different poisoning strategies. The first distinction is the class-agnostic and the class-specific backdoors [15]. The class-agnostic backdoor can be activated by a trigger injected into any input. On the other hand, the class-specific backdoor is activated only if the poisoned input belongs to a specific class. The main difference between these two strategies is that in the second case, the model needs to identify both features of the trigger and the source class making possible countermeasures more challenging [16]. Considering class-agnostic backdoor attacks, we can differentiate between the "simple" backdoor attack [19] and the clean-label backdoor attack [47].

Simple Backdoor Attack. In the rest of this paper, by the simple backdoor attack, we are referring to the data poisoning backdoor attack that was introduced in BadNets [19]. In this case, the adversary adds a small subset of poisoned samples to the training dataset. These samples have been stamped with the adversary-chosen trigger, and their label has been changed to the target class. The target class is the output of the poisoned model when the backdoor is activated.

Clean-label Backdoor Attack: The clean-label backdoor attack was introduced in [47]. This attack is similar to a simple attack, but the adversary cannot affect the label of the injected data. The reasoning behind this attack is that the poisoned training samples can be easily identified as outliers by simple filtering mechanisms or even human inspection because the original class of these samples is different from the target class. Thus, an adaptive adversary may have to poison samples only from the target class, hoping that the model identifies the trigger pattern as a class feature. This attack is still a data poisoning backdoor attack but uses a weaker adversary making the attack more challenging.

Based on the trigger, backdoors can be either static [19] or dynamic [27]. The static backdoors are activated with a trigger that has very specific characteristics. In computer vision, such a trigger could mean a specific pixel pattern or a specific position. On the other hand, the dynamic backdoors can be activated by various triggers with different characteristics.

For graph neural networks, the first backdoor attack was proposed in [65]. In this backdoor attack, a GNN classifier predicts an attacker-chosen target label for a testing graph once a predefined subgraph is injected into the testing graph. All perturbed graphs are injected with the same trigger graph. Another backdoor attack against GNNs for the graph classification task was presented in [55], but it differs from [65] in which a universal trigger graph is assumed for all the embedded graphs. This kind of backdoor attack dynamically adapts triggers to individual graphs. The adaptive trigger is optimized in both topological structure and node features. The training processes of the trigger generation function and

the backdoored GNN model are assumed as a bi-level optimization objective [14]. The authors also adapted a backtracking-based algorithm to replace a subgraph in the original graph with the adaptive trigger graph. Xu et al. [57] explored backdoor attacks on GNNs with several explainability tools. In this work, the backdoor attack is implemented with the same strategy [65] for the graph classification task. The authors also proposed a new backdoor attack strategy for the node classification task. All the above-mentioned attacks in GNNs are gray box backdoor attacks since the adversary only modifies the training dataset instead of interfering with the training of models.

METRICS

The successful backdoor attack should always be activated when the trigger is embedded into the model's input because an adversary wants to remain stealthy and interact with the poisoned model as little as possible. Additionally, the backdoor should not affect the original task when the trigger is not included in the input. When the poisoned model does not perform well on the original task, the backdoored model will 1) raise suspicions that something is wrong and 2) not be used, thus preventing the adversary's plans. As a result, to measure the success of a backdoor attack, we require two metrics: the attack success rate and the clean accuracy drop.

Attack Success Rate (ASR). The ASR shows the reliability of the attack, and it represents the number of successfully triggered backdoors from a number of poisoned inputs:

$$ASR = \frac{\sum_{i=1}^{N} F(M^*(x_i) = y_t)}{N}, \tag{8.3}$$

where M^* is the poisoned model, x_i is an input, y_t is the target class, and $F(x)$ is a function that returns 1 if x is true and 0 otherwise.

Clean Accuracy Drop (CAD). This quantity shows the backdoor's effect on the original task. It is calculated by comparing the performance of a poisoned and a clean model for clean inputs. The accuracy drop should generally be small to keep the attack stealthy.

8.3. METHODOLOGY

8.3.1. THREAT MODEL

In this work, we implement data poisoning backdoor attacks. The adversary injects a small subset of poisoned data without knowing any information about the model architecture or the training algorithm. Thus, the attack follows a gray box threat model. This threat model is realistic as current large datasets are crowdsourced [11, 2] and malicious data may go through the validation process [39]. So, an adversary could inject trigger-stamped data in such datasets that will remain unnoticed and used during training resulting in a successful backdoor attack.

In our experiments, we investigate two different attacks, the simple data poisoning attack [19] and the clean-label attack that does not alter the labels of the poisoned data [47]. For both attacks, the adversary aims to cause targeted misclassifications with a very high probability without affecting the model's performance on the original task.

8.3.2. IMAGE CLASSIFICATION

Attacks: We use two different attacks: the simple backdoor attack and the clean-label attack.

Datasets: For our image classification backdoor attacks, we use two popular image datasets: (i) CIFAR10 that consists of 60 000 32×32 color images in ten classes, with 6 000 images per class. There are 50 000 training images and 10 000 test images. (ii) Fashion-MNIST (FMNIST) [56] - a dataset of Zalando's article images consisting of a training set of 60 000 images and a test set of 10 000 images. Each image is a 28×28 gray-scale image associated with a label from ten classes.

For the CIFAR10 dataset, we split the test set in an i.i.d manner into two 5 000 sample datasets, each used for validation and test, respectively. For the FMNIST dataset, we split the training set into two different sized datasets in an i.i.d manner: the first having 50 000 samples used for training and the second having 10 000 samples for validation. With this, we have the same size of training samples for both datasets, so comparing results between these two is easier.

Features: The input features for both neural networks are the tensor of images. For the CIFAR10 dataset, each RGB image is considered as a $[3,32,32]$ shape tensor. For FMNIST, however, the images are gray-scale, so the input has only one channel ($[1,28,28]$ shape tensor). We also did the standard normalization for input values before all train, validation, and test phases.

Models: We use two models: STRIPNet [16] and ResNet [21] with nine residual blocks (ResNet-9).

Trigger: As described in [27], various triggers have been used in image classification, and all of them resulted in successful backdoor attacks. This means that the trigger shape and pattern are not crucial for the success of a backdoor attack. Thus, for our experiments, we chose a square trigger. Its pixel intensities are random values retrieved from a continuous uniform distribution (pseudorandom generator). The seed in this generator was fixed for consistency in our experiments.

8.3.3. NATURAL LANGUAGE PROCESSING

Attacks: Similar to image classification, we use simple and clean-label backdoor attacks.

Datasets: In our experiments, we used the IMDB [33] and the AG News topic classification [64] datasets. The IMDB dataset consists of 50 000 (50%/50% training/test split) movie reviews of high polarity (either positive or negative). We used 20% of the training data for validation. The AG News topic classification dataset consists of news articles belonging to four categories (world, sports, business, and science/technology). The training set consists of 120 000 samples and the testing set of 7 600 samples. Again, we used 20% of the training data for validation resulting in 96 000 and 24 000 samples for training and validation sets, respectively.

Features: The first step of our pipeline is a `TextVectorization` layer that transforms each input to a convenient form for processing as described in [27]. As the datasets are different, we used a different sequence length for each dataset. We forced the length of each sentence to be 250 words for the IMDB dataset and 197 for the AG News dataset. Additionally, we used a vocabulary of 10 000 words that proved enough for such small datasets.

Models: We used two publicly available CCN architectures. Both the first CNN [2] and the second CNN [3] use an embedding layer as their input. However, the first CNN uses a small trainable embedding of size 16, and the second uses a pre-trained GloVe embedding [38] of size 100. We want to investigate if the attack becomes more difficult when the model uses a pretrained embedding because this is more frequent in practice. Such embeddings have been trained in large corpora of text and interpret possible connections between different words more accurately. To illustrate, Google's pretrained word2vec is trained with 100 billion words from Google News, and it contains 300-dimensional vectors for 3 million words and phrases [34]. The GloVe is trained from a corpus of 6 billion words and has a vocabulary of 400 000 words [38].
Trigger: As the trigger, we used a sentence of 1 up to 4 words from the list ["trope", "everyday", "mythology", "sparkles", "ruthless"] as defined in [32]. We applied the trigger in three positions (beginning, middle, and end) to investigate whether our models are more sensitive in specific positions.

8.3.4. SPEECH RECOGNITION

Attacks: Again, we use simple and clean-label backdoor attacks.
Datasets: For this application, we used two versions of the Speech Commands dataset as described in [28]. The first version uses ten classes of the dataset and the second 30 classes. From our experiments, we excluded the samples that lasted less than one second to avoid variable-sized inputs in our pipeline resulting in 21 312 .wav files in the first case and 58 252 files in the second case. In both cases, we use 64%/16%/20% for training, validation, and testing.
Features: As our input features, we used the MFCCs of each training input. The exact hyperparameters for this calculation are described in [28].
Models: We used one CNN [32] and one LSTM [1] for our experiments.
Trigger: Our dataset's sound files are sampled at 16kHz, and according to the Nyquist-Shannon sampling theorem, the largest tone frequency that can be included in such digital signals is 8kHz. Thus, following [27], our trigger is a 7kHz tone which is a high pitch audible sound. Following the rest of the triggers tried, this trigger differs from the normal dataset samples. It lasts from 20ms to 80ms because we want to model an adversary that is as stealthy as possible. The trigger is injected in three different positions of each sound sample (beginning, middle, and end).

8.3.5. GRAPH DATA

Attacks: As described in Section 8.2.4, for graph neural networks, we utilize two backdoor attacks, i.e., AT^{I} [65] and AT^{II} [55]. The framework for AT^{I} is illustrated in Figure 8.2. In the training phase (Figure 8.2a), the attacker injects a trigger (subgraph g_t) to a subset of training graphs and changes their labels to an attacker-chosen target label. A GNN classifier is then trained using the backdoored training dataset, and such GNN is called backdoor GNN Φ_b. In the test phase (Figure 8.2b), once the test graph is injected with the same trigger graph, the backdoored GNN is likely to misclassify the testing sample to the target label. For the node classification task, we used the backdoor attack from [57].

[2] https://www.tensorflow.org/tutorials/keras/text_classification
[3] https://keras.io/examples/nlp/pretrained_word_embeddings/

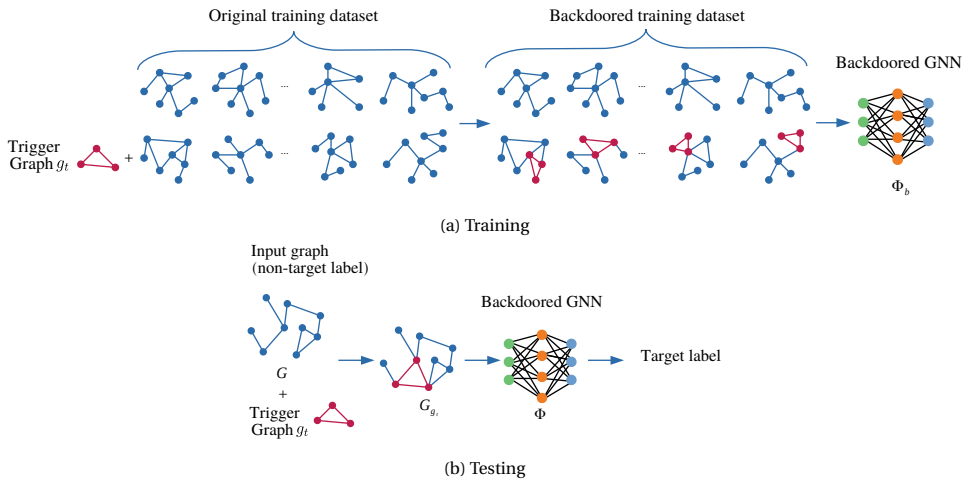


Figure 8.2: Subgraph-based backdoor attack for the graph classification task.

Since [65] and [57] designed the same strategy to implement the backdoor attack for the graph classification task, we illustrate the results of [65] and [55] for the graph classification task. The results based on [57] are presented for the node classification task.
Datasets: Table 8.1 shows the statistics for all considered datasets for graph neural networks. For the graph classification task, we use two publicly available graph datasets. (i) AIDS [35] - a dataset consisting of graphs representing molecular compounds that are active against HIV or not; (ii) TRIANGLES [35] - a synthetic dataset designed to solve the task of counting the number of triangles in a graph. For each graph classification dataset, we sample 2/3 of the graphs as the original training dataset and treat the remaining graphs as the original testing dataset. Among the original training dataset, we randomly sample α fraction of graphs to inject the trigger and relabel them with the target label, called the backdoored training dataset. Several parameters can affect the attack effectiveness: trigger size s, trigger density ρ, and poisoning intensity α. Unlike other domains, e.g., image classification, the trigger position in graph data is irrelevant and cannot be defined because a graph is non-Euclidean data where we cannot put nodes in some order. For AT^{I}, we use Erdős-Rényi (ER) model [17] to generate the trigger graph, as it is more effective than the other methods [65].

For the node classification task, we use two real-world datasets: (i) Cora [41] - a citation network in which each publication is described by a binary-valued word vector indicating the absence/presence of the corresponding word in the collection of 1433 unique words. (ii) CiteSeer [41] - another citation network with more nodes but less edges. For each node classification dataset, we split 20% of the total nodes as the original training dataset (labeled) and the rest as the original testing dataset (unlabeled). To generate the backdoored training dataset, we sample α of the original training dataset to inject the feature trigger and relabel these nodes with the target label. The feature trigger width is set to be n. Moreover, based on the conclusion in [57], different feature trigger injecting positions have a negligible impact on the attack performance, so the trigger injecting

position is randomly selected. Here, we explore the impact of poisoning intensity α and feature trigger width n on the attack performance. In the node classification task, each node feature has a value of 0 or 1, and here we set the value of the modified node features to 1 (note, the values could also be set to 0).

Table 8.1: Graph datasets statistics.

Datasets	# Graphs	Avg. # nodes	Avg. # edges	Classes	Class Distribution
AIDS	2000	15.69	16.20	2	400[0], 1600[1]
TRIANGLES	45000	20.85	32.74	10	4500[0 – 9]
Cora	1	2708	5429	7	351[0], 217[1], 418[2], 818[3], 426[4], 298[5], 180[6]
CiteSeer	1	3327	4608	6	264[0], 590[1], 668[2], 701[3], 596[4], 508[5]

Features: Each graph contains topological and node feature information. For each graph dataset in this work, there is an adjacency matrix and feature information matrix. For AIDS, Cora, and CiteSeer, there is a specific node feature vector for each node in the graph, but for TRIANGLES, the one-hot degree of a node is used as the node feature.
Models: We use two state-of-the-art GNNs for the graph classification task: GCN [26] and GraphSAGE [20]. We use GCN [26] and GAT [48] for the node classification task.
Trigger: For the graph classification task, our trigger is a global (adaptive) subgraph in AT^{I}(AT^{II}). For the node classification task, our trigger is a subset of node features with a fixed value, e.g., 0.

8.4. EXPERIMENTAL RESULTS

8.4.1. IMAGE CLASSIFICATION

Chosen settings and selected parameters. We ran our experiments with a different number of poisoned samples (25, 300, 575, 850), trigger sizes (4×4, 8×8, 12×12), and trigger positions (Upper-Mid, Mid-Left, Mid-Right, Lower-Mid) on the image. Figure 8.3 demonstrates four different positions of a 4×4 trigger for several FMNIST sample images. We repeated each experiment two times, which makes the total number of 768 experiments regarding the chosen settings. We set class number 5 as the target for all experiments and in both datasets.

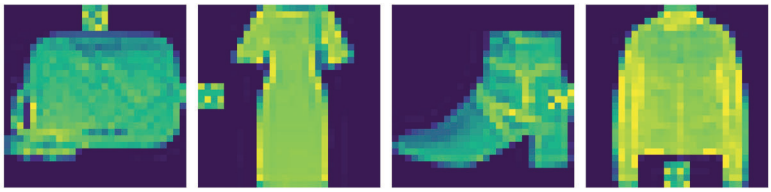


Figure 8.3: Applied 4×4 trigger in different positions: Upper-Mid, Mid-Left, Mid-Right, Lower-Mid.

Every backdoor attack should remain stealthy without affecting the original task. Therefore, the poisoned model should perform as expected when the input does not

contain the trigger. In Table 8.2, we compare the performance of clean and backdoored models for clean inputs. The attack accuracy mentioned in this table is the arithmetic mean (± the standard deviation) of the accuracy on clean inputs from all the poisoned models trained in our experiments. For the original accuracy, we trained multiple clean models and averaged their performance. The model remains unaffected from both backdoor attacks even if we use 850 poisoned samples. Such poisoning rates are small and cannot affect the model's performance in general. From Table 8.2, we can also verify that our models perform similarly well in both datasets, which is helpful when comparing the performance of the attack for each case.

Table 8.2: Clean accuracy drop in image classification.

				Number of poisoned Samples			
Dataset	Model	Original Acc	Attack Type	25	200	375	850
CIFAR10	STRIPNet	85.09 (± 0.599)	clean-label	85.22 (± 0.508)	85.38 (± 0.523)	85.37 (± 0.378)	85.23 (± 0.471)
			simple	85.31 (± 0.467)	85.28 (± 0.438)	85.35 (± 0.360)	85.12 (± 0.477)
	Resnet-9	89.99 (± 0.499)	clean-label	89.98 (± 0.356)	89.86 (± 0.300)	89.79 (± 0.461)	89.69 (± 0.377)
			simple	89.94 (± 0.353)	89.77 (± 0.274)	89.75 (± 0.266)	89.74 (± 0.378)
FMNIST	STRIPNet	93.62 (± 0.205)	clean-label	93.66 (± 0.080)	93.68 (± 0.127)	93.64 (± 0.131)	93.69 (± 0.139)
			simple	93.59 (± 0.135)	93.61 (± 0.148)	93.67 (± 0.152)	93.65 (± 0.139)
	Resnet-9	93.63 (± 0.154)	clean-label	93.59 (± 0.152)	93.53 (± 0.157)	93.62 (± 0.171)	93.59 (± 0.176)
			simple	93.57 (± 0.129)	93.55 (± 0.181)	93.59 (± 0.164)	93.58 (± 0.173)

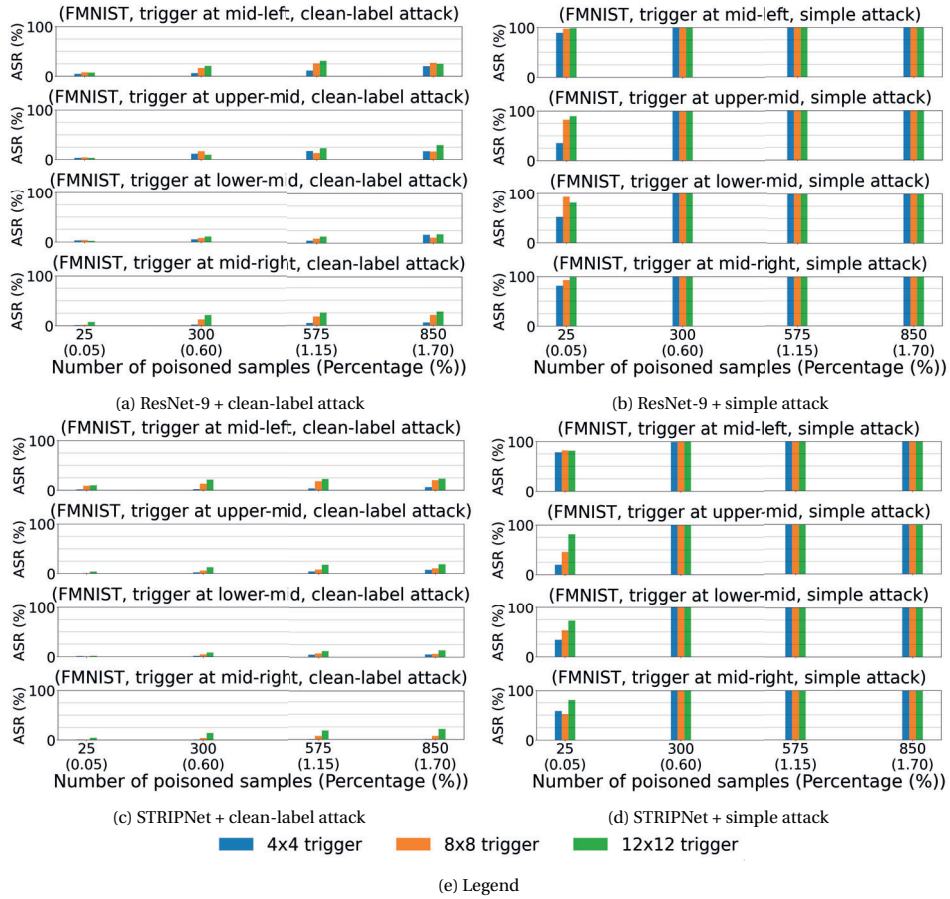
Results for FMNIST. As it can be inferred from Figure 8.4, the clean-label attack is not that effective against the FMNIST dataset. By increasing the number of poisoned samples, there are small or no improvements in attack success rate (there are small improvements when increasing the number of samples from 25 to 300, but as we increase from 300 to 575 and from 575 to 850, the improvements become even smaller). We assume this is mainly due to the dataset nature and the capability of the CNNs to learn the exclusive features of each class easily and robustly so that injecting a trigger (even of size 12×12) could not disturb the network from learning those.

Since both ResNet-9 and STRIPNet have convolutional layers, we expect negligible effects of trigger position on attack success rate. The results confirm this as there are only minor effects stemming from the trigger positions (for instance, in both networks, the trigger on the lower-mid results in the least ASR, while on the mid-right, it has a little more chance of being learned by the network. Again, we suppose this is because of the attributes of the FMNIST images and the models' focus on specific regions of an image to learn). Additionally, in almost all cases (except a few ones like upper-mid in ResNet-9), increasing the trigger size leads to higher ASR.

For the simple attack, we obtained 100% ASR for 300 attack samples or more. With 25 poisoned samples, some trigger positions have positive effects on ASR regardless of trigger size (for instance, the mid-left trigger achieves high ASR even with 4×4 size triggers).

Results for CIFAR10. The clean-label attack is significantly more effective for CIFAR10 than FMNIST (Figures 8.4 and 8.5). We believe this is primarily because the CIFAR10 images are RGB, and the crafted trigger has more layers (3 channels). As a result, the model learns the embedded trigger with less poisoned samples. As expected, the trigger position does not play an important role in ASR, and in almost all cases, ASR improves using a larger trigger size.

Another observation is that the smaller the size of the trigger, the more noticeable



(a) ResNet-9 + clean-label attack

(b) ResNet-9 + simple attack

(c) STRIPNet + clean-label attack

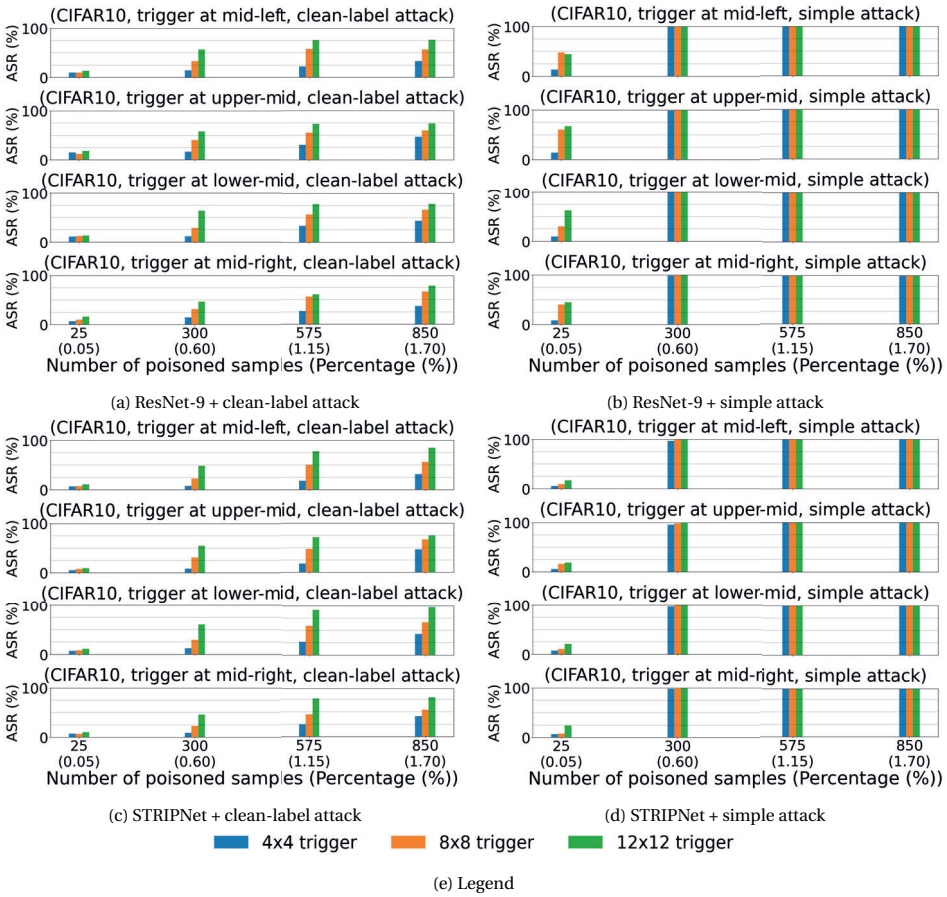
(d) STRIPNet + simple attack

(e) Legend

Figure 8.4: Attack accuracy for the FMNIST dataset. From these figures, we conclude that the clean-label attack is not effective but is slightly improved when increasing the poisoning rate. On the other hand, the simple attack can be very effective even with a small poisoning rate (0.6%). Additionally, larger triggers lead to higher ASR, but different trigger positions do not result in ASR fluctuations as the convolutional layers identify the trigger.

the ASR improvement when increasing the number of poisoned samples from 25 to 850. For instance, for a 12×12 trigger, there is no noticeable improvement in ASR when the number of poisoned samples increases from 575 to 850. On the other hand, using a 4×4 trigger, ASR's growth is easily observable between all four different poisoning rates.

Analyzing the simple attack, similarly to FMNIST, we achieved 100% ASR for 300 poisoned samples or more. Additionally, ResNet-9 is more vulnerable to backdoor attacks, particularly when using fewer poisoned samples and smaller triggers. We believe this is mostly because ResNet-9 is a larger network than STRIPNet and can extract more data from the given dataset.



(a) ResNet-9 + clean-label attack

(b) ResNet-9 + simple attack

(c) STRIPNet + clean-label attack

(d) STRIPNet + simple attack

(e) Legend

Figure 8.5: ASR for CIFAR10. The clean-label attack is significantly more effective than for FMNIST because the 3-channel trigger contains more information. We also see that the trigger position is not very important, and ASR increases as the trigger size increases. The ASR with the simple attack is 100% for a 0.6% poisoning rate or more. However, STRIPNet is not as vulnerable as ResNet due to its smaller capacity.

8.4.2. NATURAL LANGUAGE PROCESSING

In Tables 8.3 and 8.4, we compare the performance of clean and backdoored models in text classification for clean inputs. These tables are generated by averaging the performance of clean and poisoned models as described in Section 8.4.1. In all cases, the model's performance remains almost unaffected after the backdoor insertion. There are a few minor accuracy drops that are at most 0.6% making the attack stealthy. This is expected as we poison only a small subset of the training data that cannot substantially affect the model's learning.

In Figures 8.6 and 8.7, we show the results of our experiments for the AG News topic classification dataset and IMDB dataset, respectively. From these figures, we can draw several conclusions. In most cases, the ASR is correlated with the trigger size and increases as the trigger size increases. This is true even when the attack is not effective (see

Table 8.3: Clean accuracy drop in text classification (AG News).

Model	Original Acc	Attack Type	Number of poisoned samples			
			250	500	750	1000
CNN	90.74 (± 0.314)	clean-label	90.48 (± 0.557)	90.52 (± 0.531)	90.52 (± 0.614)	90.51 (± 0.541)
		simple	90.49 (± 0.534)	90.38 (± 0.593)	90.53 (± 0.484)	90.33 (± 0.760)
CNN + GloVe	89.78 (± 0.169)	clean-label	89.72 (± 0.218)	89.71 (± 0.201)	89.71 (± 0.186)	89.70 (± 0.220)
		simple	89.72 (± 0.209)	89.68 (± 0.206)	89.69 (± 0.227)	89.68 (± 0.202)

Table 8.4: Clean accuracy drop in text classification (IMDB).

Model	Original Acc	Attack Type	Number of poisoned samples			
			50	100	150	200
CNN	87.05 (± 0.062)	clean-label	86.91 (± 0.084)	86.93 (± 0.071)	86.91 (± 0.078)	86.90 (± 0.072)
		simple	86.90 (± 0.083)	86.89 (± 0.089)	86.88 (± 0.090)	86.88 (± 0.096)
CNN + GloVe	84.20 (± 0.308)	clean-label	83.73 (± 0.681)	83.71 (± 0.702)	83.62 (± 0.775)	83.77 (± 0.680)
		simple	83.79 (± 0.535)	83.82 (± 0.519)	83.61 (± 0.570)	83.59 (± 0.724)

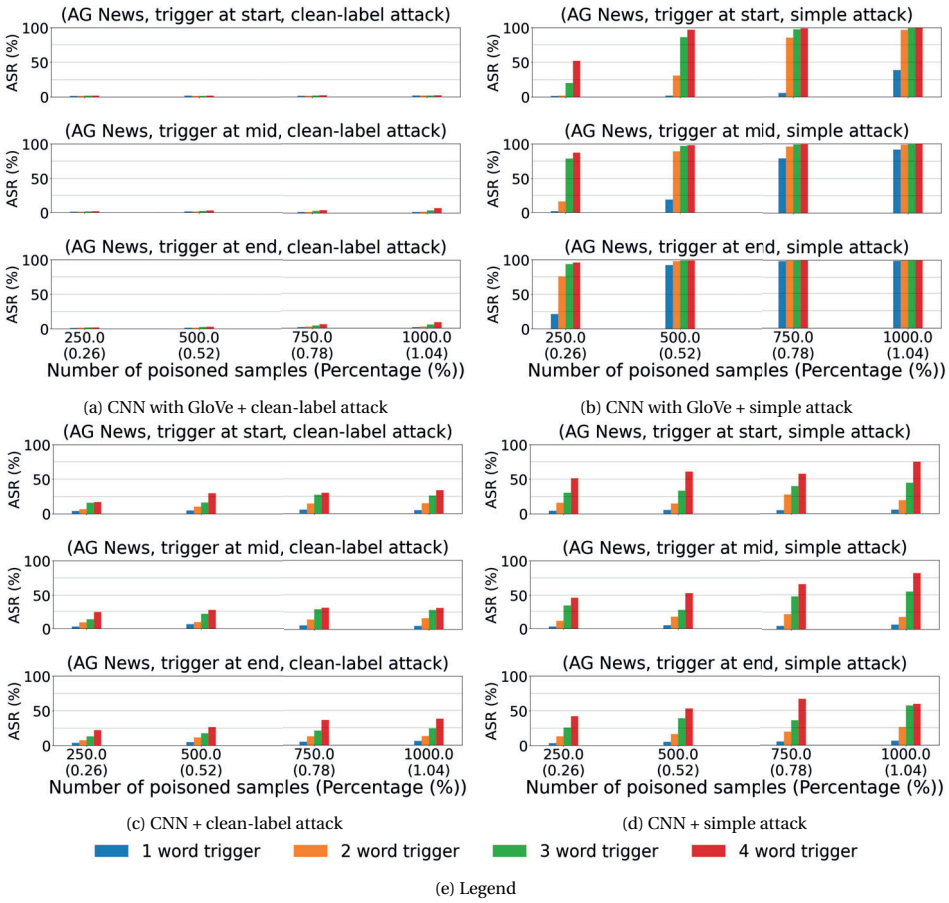
Figure 8.6a).

Especially for the first CNN, this relation seems to be linear (see Figures 8.6c, 8.6d, 8.7c and 8.7d). This simple model uses global average pooling as its penultimate layer, averaging the feature map before the output. As a result, the trigger will be more influential when it consists of more words. In almost all experiments, the poisoning rate is a highly influential hyperparameter of the backdoor attack, and any increase in it leads to an increase in the attack success rate.

Our models learn differently, which can be seen from the varying attack success rate when the trigger is injected in different positions. For example, the attack success rate is higher if the trigger is inserted at the end of the sentence when we use the first model and the simple backdoor attack (see Figures 8.6b and 8.7b). This difference is very clear for low poisoning rates (0.25%), where even a small trigger of 2 words could be substantially more effective when placed at the end of the sentence. On the other hand, for the other model, placing the trigger at the end does not result in higher ASR (see Figures 8.6d and 8.7d). These differences indicate that we could use the backdoor attack as a tool for AI explainability and further understand what and how a model learns by using triggers with different characteristics.

In [47], the authors claimed that the clean-label backdoor attack needs a very large poisoning rate to be effective. We also see this behavior in the AG News dataset, especially for the architecture that uses the pretrained GloVe embedding (Figure 8.6a). In the other architecture, the clean-label attack is more effective as the feature space created by the trainable embedding encodes some information about the trigger and the target class (see Figure 8.6c). However, when the IMDB dataset is used, both models perform similarly without poisoning a very large part of the training data for the clean-label (see Figures 8.7a and 8.7c). This can be explained by the differences between the datasets. Each sentence in AG News is shorter than the movie reviews in the IMDB dataset. Additionally, most of the words in AG News are strongly connected with the topic that each sentence belongs to (world, sports, business, and science/technology), which is not true for the IMDB dataset.

In the IMDB dataset, the sentences are longer, and usually, only a few words are related to their sentiment. As a result, in AG News, our attack needs more poisoned samples to overcome the effect of the original features of the source class.



(a) CNN with GloVe + clean-label attack

(b) CNN with GloVe + simple attack

(c) CNN + clean-label attack

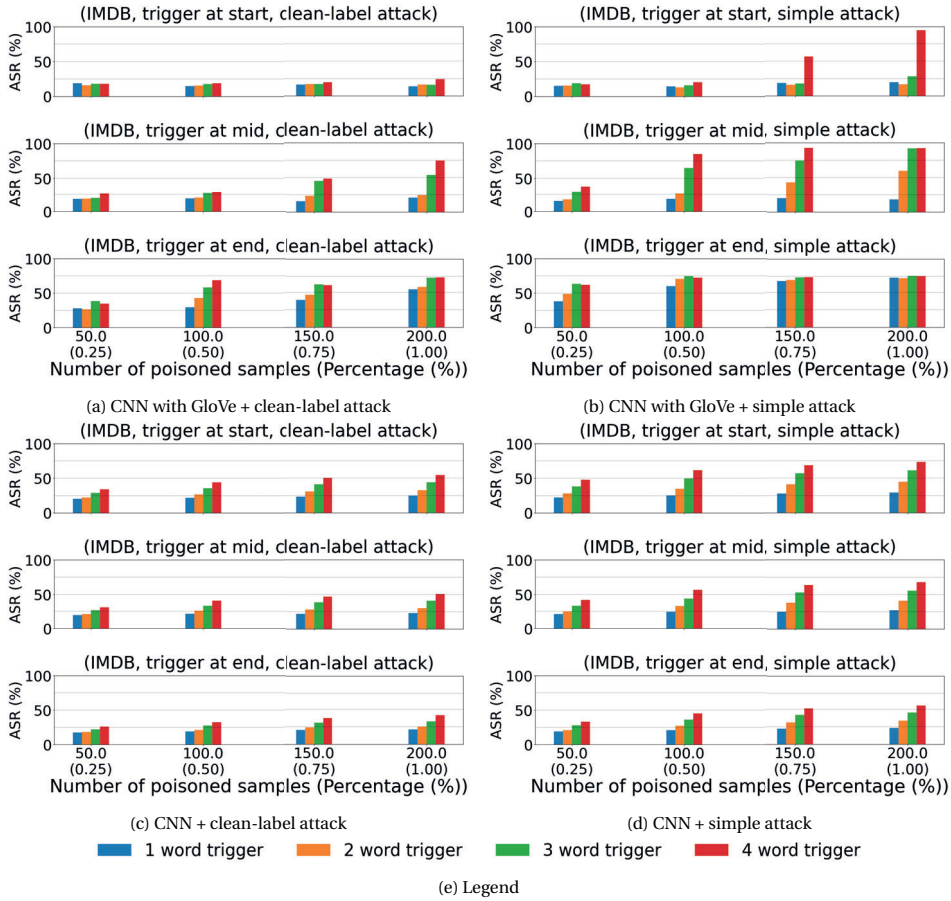
(d) CNN + simple attack

(e) Legend

Figure 8.6: Attack accuracy for the AG News dataset. The ASR is positively correlated with the trigger size (even when the ASR is very low), and the poisoning rate significantly influences the attack's effectiveness. Additionally, the clean-label attack needs more poisoned data to work. When GloVe is used, inserting the trigger in the end results in higher ASR (especially for low poisoning rates), but in the simple CNN, the trigger positions do not affect ASR.

8.4.3. SPEECH RECOGNITION

In Tables 8.5 and 8.6, we compare the performance of clean and backdoored models for sound classification when clean inputs are used. As was also shown in [28], the differences between the clean and the backdoored models are negligible. In particular, the backdoored models perform a little better when the 10 classes dataset is used, meaning that the poisoned samples could serve as a generalization factor. However, when the



(a) CNN with GloVe + clean-label attack

(b) CNN with GloVe + simple attack

(c) CNN + clean-label attack

(d) CNN + simple attack

(e) Legend

Figure 8.7: Attack accuracy for the IMDB dataset. The ASR is positively correlated with the trigger size, and the poisoning rate significantly influences the attack's effectiveness. Additionally, the clean-label attack is more effective with this dataset. For the CNN that uses GloVe, placing the trigger at the end of the sentence yields the best results, but for the simple CNN, this is the least effective position.

full dataset is used, the backdoor insertion results in a small performance drop for the CNN. In this case, we use more classes, and the model has to learn a more difficult task that is affected even by a few poisoned samples. The performance of the LSTM is slightly increased, meaning that the LSTM builds different models and utilizes its capacity better when we use the full dataset [28]. All these differences are small, and our claims need additional experimental data to be confirmed.

In Figures 8.8 and 8.9, we show the results of our experiments for the first (10 classes) and the second (30 classes) version of the Speech Commands dataset. In almost all cases, the attack success rate increases as the trigger duration increases. This is true even when the attack is not successful (see the clean-label attack in Figures 8.8c and 8.9c). This makes sense as more input features are affected when a longer trigger is used, and the

Table 8.5: Clean accuracy drop in sound classification (10 classes).

			Number of poisoned samples			
Model	Original Acc	Attack Type	50	100	150	200
CNN	94.82 (± 0.360)	clean-label	95.07 (± 0.437)	95.00 (± 0.477)	95.07 (± 0.427)	94.96 (± 0.508)
		simple	95.09 (± 0.417)	95.04 (± 0.438)	94.99 (± 0.474)	94.95 (± 0.480)
LSTM	89.47 (± 1.412)	clean-label	89.77 (± 1.426)	89.69 (± 1.350)	90.05 (± 1.362)	89.89 (± 1.373)
		simple	89.71 (± 1.605)	89.96 (± 1.335)	89.66 (± 1.482)	89.80 (± 1.586)

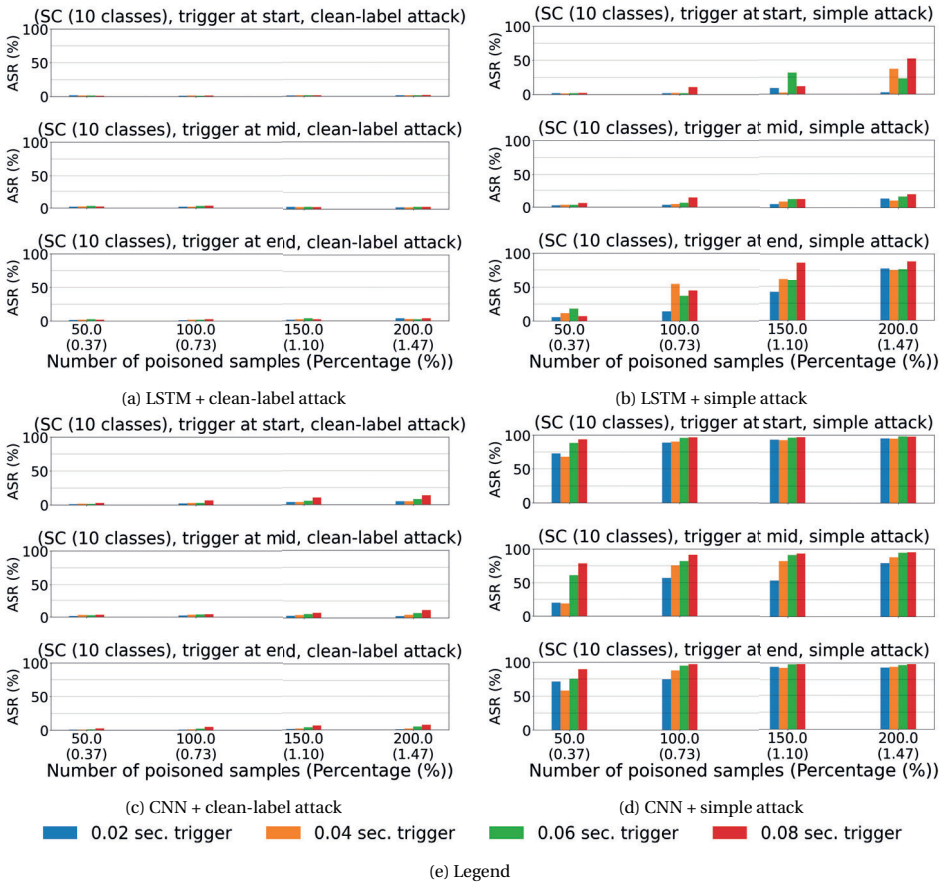
Table 8.6: Clean accuracy drop in sound classification (30 classes).

			Number of poisoned samples			
Model	Original Acc	Attack Type	137	274	411	547
CNN	94.70 (± 0.395)	clean-label	94.61 (± 0.410)	94.55 (± 0.439)	94.49 (± 0.575)	94.59 (± 0.461)
		simple	94.54 (± 0.471)	94.63 (± 0.511)	94.56 (± 0.496)	94.53 (± 0.477)
LSTM	90.50 (± 0.967)	clean-label	90.88 (± 1.267)	90.59 (± 1.238)	90.82 (± 1.232)	90.79 (± 1.334)
		simple	90.86 (± 1.236)	90.81 (± 1.167)	90.67 (± 1.317)	90.63 (± 1.278)

network can learn this relation easier. Additionally, the poisoning rate is very influential, and its increase leads to more effective backdoors.

The end of the input is the most effective trigger position for the LSTM network in both versions of the dataset. Even though this network uses two bidirectional LSTM layers and an attention layer, it seems to learn the features that are placed towards the end of its inputs more easily. The LSTM network was designed to tackle the problem of long-term dependencies on its inputs. A possible reason for this behavior is the nature of this particular dataset, which consists of 1-second clips of spoken words. If these words are not perfectly centered and distributed to the upper half of each sample, our network will give more attention to the end of each training sample. This is not true for the CNN used as, in that case, all the positions seem to be equally effective (see Figures 8.8d and 8.9d). Similarly to text classification, different models learn different patterns from the same dataset making the backdoor attack effective in different cases. Thus, we could use the backdoor attack and its triggers to understand what a model learns and how it makes its decisions.

In our sound classification experiments, the clean-label attack is not successful for both neural networks and datasets. However, when the full dataset and CNN are used (Figure 8.9c), the attack success rate slightly increases with large triggers. The clean-label could work without requiring more poisoned data if we choose a larger trigger. This claim, though, needs to be verified in the future with more experimental evidence. Another interesting observation is that the simple backdoor attack becomes more effective when we use the full dataset and the LSTM network (see Figure 8.9b). One reason for this behavior is the absolute number of training samples that were increased when the full dataset was used. However, this should be investigated further.



(a) LSTM + clean-label attack

(b) LSTM + simple attack

(c) CNN + clean-label attack

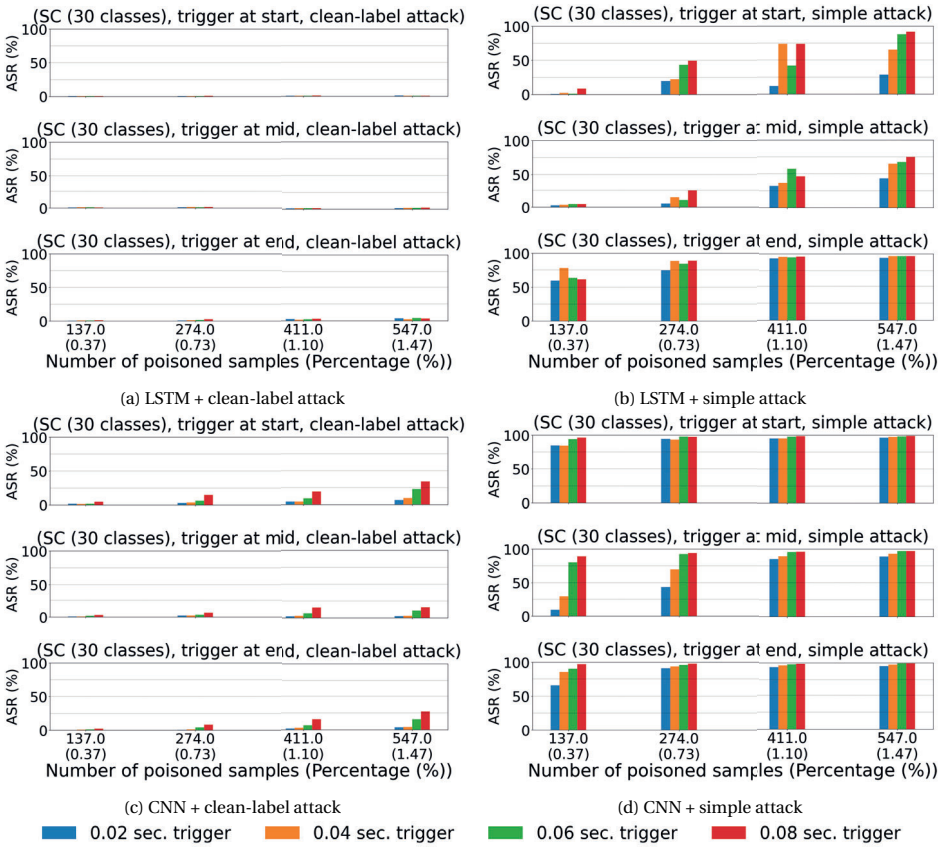
(d) CNN + simple attack

(e) Legend

Figure 8.8: Attack accuracy for the Speech Commands dataset (10 classes). In most cases, ASR is increased as the trigger duration or the poisoning rate increase. The clean-label attack is ineffective for both models. For LSTM, the best position for the trigger is at the end. However, for CNN, any position works. For CNN, the simple attack works almost perfectly for 100 poisoned samples or more.

8.4.4. GRAPH DATA

Results for the graph classification task. For the graph classification task, two parameters affect the performance of the backdoor attack: poisoning intensity and trigger size (the number of nodes in the trigger graph). The attack results for the GCN model on AIDS with different poisoning intensity α and trigger size s are shown in Figure 8.10. As we can see from Figure 8.10a, with the increase of poisoning intensity, the attack success rate is generally increasing for each trigger size, but there is no obvious improvement between $\alpha = 0.15$ and $\alpha = 0.2$. Here, we select poisoning intensity $\alpha = 0.15$ for GCN on AIDS. Figure 8.10b shows the impact of trigger size under the selected poisoning intensity ($\alpha = 0.15$). The attack success rate is highest with $s = 5$, while the clean accuracy drop is the smallest when $s = 5$. To compare the two backdoor attacks, we set $\alpha = 0.15$, $s = 5$ and $\alpha = 0.2$, $s = 7$ for AIDS and TRIANGLES, respectively.



(a) LSTM + clean-label attack

(b) LSTM + simple attack

(c) CNN + clean-label attack

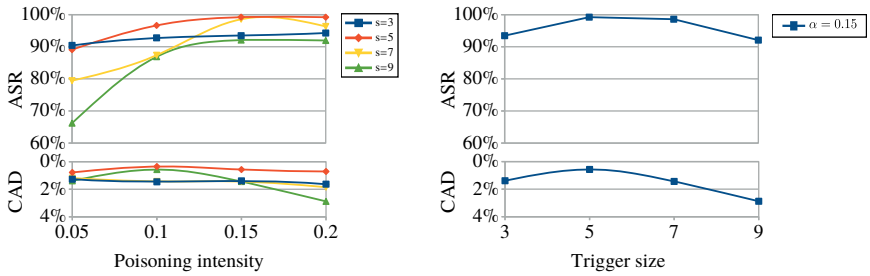
(d) CNN + simple attack

(e) Legend

Figure 8.9: Attack accuracy for the Speech Commands dataset (30 classes). In most cases, ASR is increased as the trigger duration or the poisoning rate increase. The clean-label attack is ineffective in that case, but it works slightly better for CNN. In the simple attack, the best position for the trigger is at the end for LSTM. However, there is no difference for CNN (in most cases, ASR is close to 100%). When using LSTM, ASR is higher than ASR for the ten classes. We assume that the absolute number of poisoned samples could be the reason behind that behavior.

Specifically, we present the attack results of two backdoor attacks on the graph classification task in Tables 8.7 and 8.8. As we can see from Table 8.7, AT^{II} can achieve more than 99% attack success rate and less than 1% clean accuracy drop on AIDS, while the performance of AT^{I} degrades slightly with an attack success rate of more than 95% and clean accuracy drop around 1.5%. As illustrated in Table 8.8, the attack success rate of AT^{II} is significantly higher than AT^{I} for TRIANGLES, i.e., more than 10%. However, the clean accuracy drop of AT^{II} is larger than AT^{I}, which is more than 4% for both models, while that of AT^{I} is around 3% and less than 1% for GCN and GraphSAGE, respectively. In addition, the computation time for AT^{II} is around 1.7 times of AT^{I}.

Results for the node classification task. For the node classification task, the backdoored



(a) GCN_AIDS (b) GCN_AIDS ($\alpha = 0.15$)

Figure 8.10: Impact of poisoning intensity and trigger size on attack performance in the graph classification task.

Table 8.7: Backdoor attack results for the graph classification task and the AIDS dataset.

Setting	AT^{I}		AT^{II}	
	ASR (%)	CAD (%)	ASR (%)	CAD (%)
GCN	95.86	1.25	99.92	0.46
GraphSAGE	97.59	1.46	99.80	0.91

Table 8.8: Backdoor attack results for the graph classification task and the TRIANGLES dataset.

Setting	AT^{I}		AT^{II}	
	ASR (%)	CAD (%)	ASR (%)	CAD (%)
GCN	86.00	3.18	99.21	5.32
GraphSAGE	87.70	0.50	98.24	4.32

data is influenced by two parameters: poisoning intensity α and feature trigger width n. The attack performance, including attack success rate and clean accuracy drop with different variants, is shown in Figure 8.11. For each feature trigger width, the attack success rate on different models and datasets generally increases when the poisoning intensity increases from 0.05 to 0.2. At the same time, the clean accuracy drop of the GCN model keeps increasing, and there is a significant increase between $\alpha = 0.15$ and $\alpha = 0.2$. However, the clean accuracy drop of the GAT model remains almost unchanged. To achieve a high attack success rate and low clean accuracy drop, we set $\alpha = 0.2$ for GCN and $\alpha = 0.15$ for GAT. To evaluate the impact of feature trigger width on attack performance, we show the attack results with different feature trigger widths in Figure 8.12. Observe that the feature trigger width has no obvious influence on the attack success rate and clean accuracy drop for both GNN models and datasets.

Specifically, Table 8.9 shows the attack success rate and clean accuracy drop of backdoor attack for the node classification task with selected parameters. Notice that the backdoor attack on GCN reaches over 70% attack success rate for both datasets and that on GAT obtains a higher attack success rate, i.e., over 85% and 90% for Cora and CiteSeer,

respectively. Furthermore, the clean accuracy drop is lower than 2% for all models and datasets except for the GAT model on the Cora dataset, which is 2.35%.

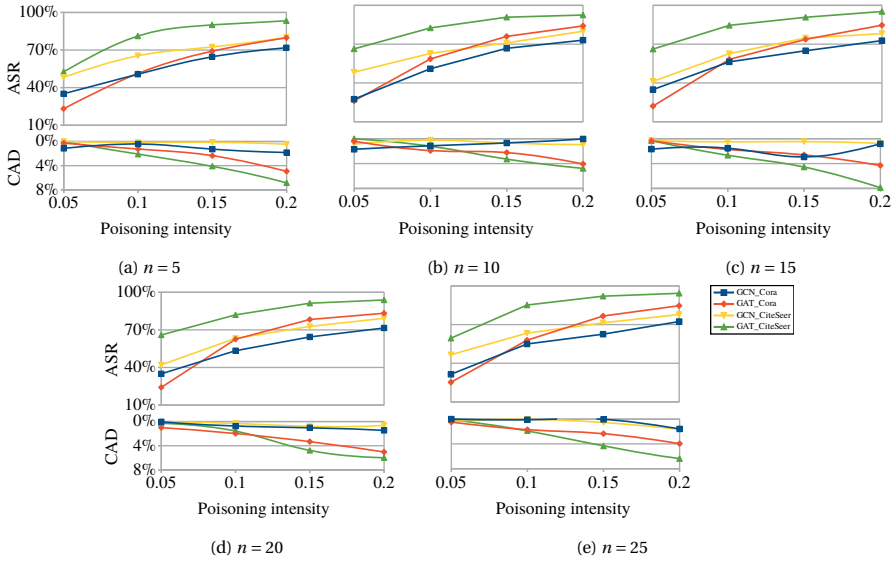


Figure 8.11: Impact of poisoning intensity and feature trigger width on attack performance in the node classification task.

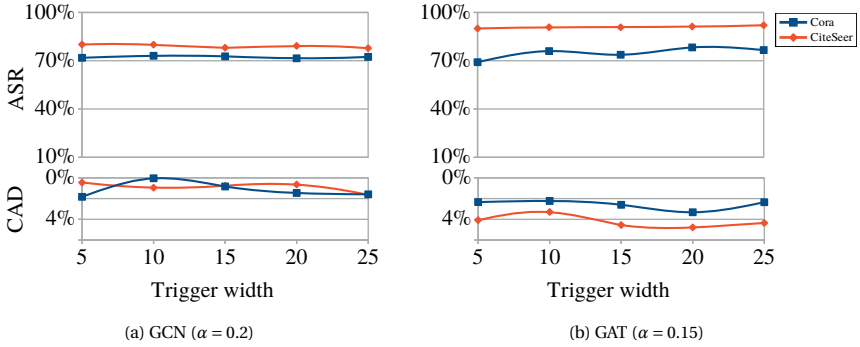


Figure 8.12: Attack performance with different feature trigger widths.

8.4.5. General Observations

First, we verified that the backdoor attack is a real threat as it can be injected into every application domain tried without affecting the model's original task just by poisoning a small subset of the training data. Additionally, we saw that the poisoning rate is the most influential characteristic of the trigger in all applications. However, this value cannot be increased arbitrarily because the backdoor attack will become evident through a simple

Table 8.9: Backdoor attack results in the node classification task ($n = 5$).

Setting	GCN ($\alpha = 0.2$)		GAT ($\alpha = 0.15$)	
	ASR (%)	CAD (%)	ASR (%)	CAD (%)
Cora	72.35	1.59	86.63	2.35
CiteSeer	77.82	1.63	92.04	1.35

data filtering mechanism, and the poisoned model's performance on clean inputs will decrease substantially.

The trigger size is positively correlated with the backdoor's attack success rate in image, text, and sound. This is expected as a larger trigger contains more information that can be encoded easier in the trained model. However, in graph classification, the attack success rate increased to a point ($s = 5$) and then decreased for larger triggers. The variations are small, though, as ASR remained above 90% in our experiments, and thus, we cannot draw general conclusions. We need to verify this effect with more complex datasets and models.

The most effective position of the trigger (if there is any) depends on many factors, like the network architecture or the dataset. The position is not very influential on the attack success rate in most cases, but this is not always true. Thus, we cannot draw any general conclusions. In image classification, no position was proven more effective as the convolutional layers extract information from any point in the image. Similar behavior has been observed in graph neural networks [57], where the trigger position did not result in more effective backdoors. On the other hand, in text classification, the attack performed similarly for all the trigger positions for the simple CNN, but the "end" was slightly more effective when the GloVe embedding was used. In sound classification, the trigger was more effective in the end if LSTM was used but had no difference for CNN. These differences suggest a potential beneficial use case for backdoor attacks in general. In this case, we can use them to understand better how and what our models learn. Such an approach complements the work described in [61], where the authors drew valuable insights about the input's crucial features after graying out small square areas of the input images.

The clean-label attack is challenging in image, text, and sound classification. However, in some cases, it may be successful just by using a large trigger without having to poison more data. Additionally, if the trigger encloses more information, the clean-label's performance can be improved. We verified this for the CIFAR10 dataset, where we injected our trigger in all three image channels. We believe that the dataset influences the performance of this attack. If each element contains many features, the model will require a large poisoning rate to perceive the trigger as a feature of this class. In the clean-label attack, the trigger is injected only in elements from the target class, and it is not easy to overcome the effect of the actual features of this class. This was highlighted in the AG News and IMDB datasets in text classification. On the other hand, the simple backdoor attack can be very effective with just a few poisoned samples in all the applications we tried.

As a general remark, we believe that the backdoor is easier inserted into models that

can overfit small subsets of their datasets. Models with strong generalizations are more robust against data poisoning backdoor attacks. We verified this behavior using simple CNN in text classification. In that case, our attack could not reach an attack success rate larger than 80% even with the simple attack, as the model is very simple and the learned function is very smooth. Finally, as our experiments are far from exhaustive, our findings should be taken as indications, not definitive conclusions.

In summary, the key takeaways are:

- The backdoor attack is a realistic and stealthy threat.
- As expected, increasing the poisoning rate and using larger triggers leads to higher ASR.
- Different models can behave differently during the attack even though we use the same data. Therefore, we can use the backdoor attacks as a tool for explainable AI.
- In [47], the authors claimed that the clean-label attack is not very effective. In most cases, this is true, but we saw that we could make it effective with more sophisticated triggers.
- The backdoor is easier for models that can overfit a small subset of their datasets.

8.5. CONCLUSIONS

Recent trends in machine learning lead to novel attack vectors like the backdoor attack. This attack is very dangerous as it can compromise AI-powered systems. Naturally, the backdoor attack also attracted significant attention, resulting in numerous novel attack and defense versions. In this work, we explored the effects of various trigger characteristics on the backdoor's performance in four domains. Our results show that deploying backdoor attacks is relatively easy for all investigated domains. There are sufficient commonalities between the attacks in different domains to ease their deployment in real-world applications and devise novel, more generic defenses.

REFERENCES

[1] Douglas Coimbra de Andrade et al. *A neural attention model for speech command recognition*. 2018. arXiv: 1808.08929 [eess.AS].

[2] Rosana Ardila et al. "Common Voice: A Massively-Multilingual Speech Corpus". In: *CoRR* abs/1912.06670 (2019). arXiv: 1912.06670. URL: http://arxiv.org/abs/1912.06670.

[3] Eugene Bagdasaryan and Vitaly Shmatikov. "Blind Backdoors in Deep Learning Models". In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 1505–1521. ISBN: 978-1-939133-24-3. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/bagdasaryan.

[4] Eugene Bagdasaryan et al. "How to backdoor federated learning". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 2938–2948.

[5] Bryant Chen et al. *Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering*. 2018. arXiv: 1811.03728 [cs.LG].

[6] Xiaoyi Chen et al. "Badnl: Backdoor attacks against nlp models". In: *ICML 2021 Workshop on Adversarial Machine Learning*. 2021.

[7] Xinyun Chen et al. "Targeted backdoor attacks on deep learning systems using data poisoning". In: *arXiv preprint arXiv:1712.05526* (2017).

[8] Robby Costales et al. "Live Trojan attacks on deep neural networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 796–797.

[9] George E Dahl et al. "Large-scale malware classification using random projections and neural networks". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2013, pp. 3422–3426.

[10] Jiazhu Dai, Chuanshuai Chen, and Yufeng Li. "A backdoor attack against lstm-based text classification systems". In: *IEEE Access* 7 (2019), pp. 138872–138878.

[11] J. Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[12] Murat Dikmen and Catherine M Burns. "Autonomous driving in the real world: Experiences with tesla autopilot and summon". In: *Proceedings of the 8th international conference on automotive user interfaces and interactive vehicular applications*. 2016, pp. 225–228.

[13] Samuel Dodge and Lina Karam. "A study and comparison of human and deep learning recognition performance under visual distortions". In: *2017 26th international conference on computer communication and networks (ICCCN)*. IEEE. 2017, pp. 1–7.

[14] Luca Franceschi et al. "Bilevel programming for hyperparameter optimization and meta-learning". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1568–1577.

[15] Yansong Gao et al. "Backdoor attacks and countermeasures on deep learning: A comprehensive review". In: *arXiv preprint arXiv:2007.10760* (2020).

[16] Yansong Gao et al. "STRIP: A Defence against Trojan Attacks on Deep Neural Networks". In: *Proceedings of the 35th Annual Computer Security Applications Conference*. ACSAC '19. San Juan, Puerto Rico, USA: Association for Computing Machinery, 2019, pp. 113–125. ISBN: 9781450376280. DOI: 10.1145/3359789.3359790. URL: https://doi.org/10.1145/3359789.3359790.

[17] E. N. Gilbert. "Random Graphs". In: *The Annals of Mathematical Statistics* 30.4 (1959), pp. 1141–1144. DOI: 10.1214/aoms/1177706098. URL: https://doi.org/10.1214/aoms/1177706098.

[18] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks". In: *2013 IEEE international conference on acoustics, speech and signal processing*. Ieee. 2013, pp. 6645–6649.

[19] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. "Badnets: Identifying vulnerabilities in the machine learning model supply chain". In: *arXiv preprint arXiv:1708.06733* (2017).

[20] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs". In: *Advances in neural information processing systems* 30 (2017).

[21] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[22] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. "Handcrafted backdoors in deep neural networks". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 8068–8080.

[23] IBM. *Natural Language Processing*. https://www.ibm.com/cloud/learn/natural-language-processing. [Online, Accessed: 27 July 2022].

[24] Simen Skaret Karlsen. "Automated Front Detection-Using computer vision and machine learning to explore a new direction in automated weather forecasting". MA thesis. The University of Bergen, 2017.

[25] Asharul Islam Khan and Salim Al-Habsi. "Machine learning in computer vision". In: *Procedia Computer Science* 167 (2020), pp. 1444–1451.

[26] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *ICLR*. 2017.

[27] Stefanos Koffas, Stjepan Picek, and Mauro Conti. "Dynamic Backdoors with Global Average Pooling". In: *arXiv preprint arXiv:2203.02079* (2022).

[28] Stefanos Koffas et al. "Can You Hear It? Backdoor Attacks via Ultrasonic Triggers". In: *Proceedings of the 2022 ACM Workshop on Wireless Security and Machine Learning*. WiseML '22. San Antonio, TX, USA: Association for Computing Machinery, 2022, pp. 57–62. ISBN: 9781450392778. DOI: 10.1145/3522783.3529523. URL: https://doi.org/10.1145/3522783.3529523.

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Commun. ACM* 60.6 (May 2017), pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386. URL: https://doi.org/10.1145/3065386.

[30] Shaofeng Li et al. "Invisible backdoor attacks on deep neural networks via steganography and regularization". In: *IEEE Transactions on Dependable and Secure Computing* 18.5 (2020), pp. 2088–2105.

[31] Yiming Li et al. "Backdoor learning: A survey". In: *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[32] Yingqi Liu et al. "Trojaning Attack on Neural Networks". In: *25nd Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-221, 2018*. The Internet Society, 2018.

[33] Andrew L. Maas et al. “Learning Word Vectors for Sentiment Analysis”. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. URL: http://www.aclweb.org/anthology/P11-1015.

[34] Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).

[35] Christopher Morris et al. “TUDataset: A collection of benchmark datasets for learning with graphs”. In: *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*. 2020. arXiv: 2007.08663. URL: www.graphlearning.io.

[36] Nurulain Abd Mubin et al. “Young and mature oil palm tree detection and counting using convolutional neural network deep learning method”. In: *International Journal of Remote Sensing* 40.19 (2019), pp. 7500–7515.

[37] Blaine Nelson et al. “Misleading Learners: Co-opting Your Spam Filter”. In: *Machine Learning in Cyber Trust: Security, Privacy, and Reliability*. Boston, MA: Springer US, 2009, pp. 17–51. ISBN: 978-0-387-88735-7. DOI: 10.1007/978-0-387-88735-7_2. URL: https://doi.org/10.1007/978-0-387-88735-7%5C_2.

[38] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. “GloVe: Global Vectors for Word Representation”. In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[39] Vinay Uday Prabhu and Abeba Birhane. “Large image datasets: A pyrrhic win for computer vision?” In: *CoRR* abs/2006.16923 (2020). arXiv: 2006.16923. URL: https://arxiv.org/abs/2006.16923.

[40] Jorai Rijsdijk et al. “Reinforcement Learning for Hyperparameter Tuning in Deep Learning-based Side-channel Analysis”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2021.3 (July 2021), pp. 677–707. DOI: 10.46586/tches.v2021.i3.677-707. URL: https://tches.iacr.org/index.php/TCHES/article/view/8989.

[41] Prithviraj Sen et al. “Collective classification in network data”. In: *AI magazine* 29.3 (2008), pp. 93–93.

[42] Giorgio Severi et al. “Explanation-Guided Backdoor Poisoning Attacks Against Malware Classifiers”. In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 1487–1504. ISBN: 978-1-939133-24-3. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/severi.

[43] Reza Shokri et al. “Bypassing backdoor detection algorithms in deep learning”. In: *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2020, pp. 175–183.

[44] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *nature* 529.7587 (2016), pp. 484–489.

[45] Ziteng Sun et al. *Can You Really Backdoor Federated Learning?* 2019. arXiv: 1911.07963 [cs.LG].

[46] Paulo Trigueiros, Fernando Ribeiro, and Luís Paulo Reis. "Hand gesture recognition system based in computer vision and machine learning". In: *Developments in Medical Image Processing and Computational Vision*. Springer, 2015, pp. 355–377.

[47] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. "Label-consistent backdoor attacks". In: *arXiv preprint arXiv:1912.02771* (2019).

[48] Petar Veličković et al. "Graph Attention Networks". In: *ICLR* (2018). URL: https://openreview.net/forum?id=rJXMpikCZ.

[49] Silvia Vinyes Mora. "Computer vision and machine learning for in-play tennis analysis: framework, algorithms and implementation". PhD thesis. Imperial College London, 2018.

[50] B. Wang et al. "Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks". In: *2019 IEEE Symposium on Security and Privacy (SP)*. 2019, pp. 707–723.

[51] Haiyan Wang et al. "An end-to-end traffic vision and counting system using computer vision and machine learning: the challenges in real-time processing". In: *SIGNAL 2018 Editors* (2018), p. 13.

[52] Emily Wenger et al. "Backdoor Attacks Against Deep Learning Systems in the Physical World". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 6206–6215.

[53] Victor Wiley and Thomas Lucas. "Computer vision and image processing: a paper review". In: *International Journal of Artificial Intelligence Research* 2.1 (2018), pp. 29–36.

[54] Yonghui Wu et al. "Google's neural machine translation system: Bridging the gap between human and machine translation". In: *arXiv preprint arXiv:1609.08144* (2016).

[55] Zhaohan Xi et al. "Graph backdoor". In: *30th USENIX Security Symposium (USENIX Security 21)*. 2021, pp. 1523–1540.

[56] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. Aug. 28, 2017. arXiv: cs.LG/1708.07747 [cs.LG].

[57] Jing Xu, Minhui Xue, and Stjepan Picek. "Explainability-based backdoor attacks against graph neural networks". In: *Proceedings of the 3rd ACM Workshop on Wireless Security and Machine Learning*. 2021, pp. 31–36.

[58] Zhaoyuan Yang et al. "Design of intentional backdoors in sequential models". In: *arXiv preprint arXiv:1902.09972* (2019).

[59] Zhitao Ying et al. "Hierarchical graph representation learning with differentiable pooling". In: *Advances in neural information processing systems* 31 (2018).

[60] Gu Yunchao and Yang Jiayao. "Application of computer vision and deep learning in breast cancer assisted diagnosis". In: *Proceedings of the 3rd International Conference on Machine Learning and Soft Computing*. 2019, pp. 186–191.

[61] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[62] Tongqing Zhai et al. "Backdoor attack against speaker verification". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 2560–2564.

[63] Muhan Zhang et al. "An end-to-end deep learning architecture for graph classification". In: *Thirty-second AAAI conference on artificial intelligence*. 2018.

[64] Xiang Zhang, Junbo Zhao, and Yann LeCun. "Character-level convolutional networks for text classification". In: *Advances in neural information processing systems* 28 (2015).

[65] Zaixi Zhang et al. "Backdoor attacks to graph neural networks". In: *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*. 2021, pp. 15–26.

9

BACKDOOR PONY: EVALUATING BACKDOOR ATTACKS AND DEFENSES IN DIFFERENT DOMAINS

Outsourced training and crowdsourced datasets lead to a new threat for deep learning models: the backdoor attack. In this attack, the adversary inserts a secret functionality in a model, activated through malicious inputs. Backdoor attacks represent an active research area due to diverse settings where they represent a real threat. Still, there is no framework to evaluate existing attacks and defenses in different domains. Only a few toolboxes have been implemented, but most of them focus on computer vision and are difficult to use. To bridge this gap, we implement Backdoor Pony, a framework for evaluating attacks and defenses in different domains through a user-friendly GUI.

This chapter has been published as A. Mercier, N. Smolin, O. Sihlovec, S. Koffas, S. Picek, "Backdoor Pony: Evaluating backdoor attacks and defenses in different domains" *SoftwareX, Volume 22, 2023, 101387, ISSN 2352-7110.*

9.1. MOTIVATION AND SIGNIFICANCE

Neural Networks have proven to work more accurately and considerably faster than humans in several areas, such as facial recognition [47], stock market predictions [8], and detection of "fake news" on social media [71]. For this reason, they are widely used and recently deployed on critical applications like autonomous driving [4]. Thus, we must ensure their secure operation and deployment in the real world. The backdoor attack is a threat that has recently emerged and can have dire consequences as the trained model can be manipulated to the adversary's benefit [21]. It is, therefore, essential to develop defenses to identify and counter these attacks. Up to date, only a few tools facilitate the evaluation of such attacks and defenses. However, most of them target only computer vision [34, 63, 46], or only offer a command line interface [46, 44]. As such, they are not convenient for a broader audience.

To bridge this gap, we implemented Backdoor Pony.[1] Backdoor Pony is a web-based application containing a GUI through which users can evaluate backdoor attacks and defenses in different application domains. Its GUI is user-friendly, making research more accessible to a large audience. The user controls various hyperparameters for the attacks and defenses through the GUI without the need to build complex pipelines. After each experiment, various plots are shown to help users understand the performance of their experiments. Our tool can follow the recent developments in this field, as it can be extended with new attacks and defenses in all domains. To conclude, our tool could lower the barrier for new researchers and establish a common point of reference for backdoor attacks and defenses.

The main difference between Backdoor Pony and the state-of-the-art can be seen in Table 9.1. We see that only OpenBackdoor [14] targets text applications, and the rest are focused only on computer vision [34, 46, 63, 59]. Backdoor Pony is the first tool to include audio and graph data. It is also the first to offer a GUI. While the existing tools have implemented more attacks and defenses, we do not consider this a major limitation, as we can use our extension API to extend our tool.

Table 9.1: A comparison of Backdoor Pony and the state-of-the-art.

Tool	Image	Text	Audio	Graph	GUI	#Attacks	#Defenses
BackdoorBox [34]	x					12	7
TrojanZoo [46]	x					8	14
OpenBackdoor [14]		x				12	4
BackdoorBench [63]	x					8	9
Backdoor Toolbox [59]	x					13	10
Backdoor Pony	x	x	x	x	x	6	5

The implemented attacks and defenses by the state-of-the-art are shown in Table 9.2 and Backdoor Pony's supported attacks and defenses in Table 9.3. In image classification, we chose patch-based triggers as they are the most popular. In particular, we chose the Badnet [21] and the clean-label attack [1]. From Table 9.2 we see that these attacks are widely supported by the state-of-the-art. Even though it is very popular, we did not use the Blended attack [11] as its performance is very similar to the Badnet attack. We chose only three different defenses that target different parts of the pipeline (input

[1] https://github.com/Rezonansce/backdoorponyv2

sanitization, backdoor removal, or monitoring) to avoid overlaps in our evaluations. For text classification, we implemented BadNL attack [9] as it supports various triggers (character, word, or sentence). As a defense, we chose ONION [51] as it is one of the first defenses for textual backdoors. In graph and audio domains, we picked the most popular attacks from the literature. In the graph domain, to the best of our knowledge, there is not yet a domain-specific defense, and for audio, we chose STRIP-VITA [19]. Our tool supports fewer defenses and attacks as we initially aimed for usability and application variety. Now that many domains are supported, we can easily extend our tool with additional attacks and defenses through our extension API.

Table 9.2: The implemented attacks and defenses by the state-of-the-art

Tool	Type	Attack	Defense
BackdoorBox	image	Badnets [21], Blended [11], Refool [38], Clean-label [1], TUAP [80], SleeperAgent [56], ISSBA [35], WaNet [42], Blind [2], Input-aware [43], PhysicalBA [33], LIRA [17]	3-layer Autoencoder [39], ShrinkPad [33], Fine-tuning, Fine-Pruning [36], MCR [79], NAD [32], ABL [31]
TrojanZoo	image	Badnets [21], ESB [58], TNN [37], Refool [38], Blended [11], LB [72], ABE [55], IMC [45]	Randomized Smoothing [13], DownUp [67], Adversarial Retraining [40], Manifold-projection [41], AC [5], Spectral-signatures [60], STRIP [20], NEO [61], Fine-Pruning [36], Neural Cleanse [62], DeepInspect [7], TABOR [22], Neuron Inspect [26], ABS
OpenBackdoor	text	Badnets [21], AddSent [15], SynBkd [48], StyleBkd [50], POR [54], TrojanLM [76], SOS [70], LWP [30], EP [68], NeuBA [78], LWS [52], RIPPLES [28]	ONION [51], STRIP-VITA [19], RAP [69], BKI [6]
BackdoorBench	image	Badnets [21], Blended [11], Clean-label [1], SIG [3], Low-frequency [74], ISSBA [35], Input-aware [43], WaNet [42]	Fine-tuning, Fine-pruning [36], NAD [32], Neural Cleanse [62], ANP [64], ABL [31], DBD [25] Activation Clustering [5], Spectral-signatures [60],
Backdoor Toolbox	image	Badnets [21], Badnets all-to-all [21], Blended [11], TNN [37], Clean-label [1], Input-aware [43], SIG [3], ISSBA [35], WaNet [42], Refool [38], TaCT [57], Adaptive [53], SleeperAgent [56]	SCAn [57], Activation clustering [5], Spectral-signatures [60], SPECTRE [23], STRIP [20], Sentinet [12], Low-frequency [74], Neural Cleanse [62], Fine-pruning [36], ABL [31]

Table 9.3: The implemented attacks and defenses per data type.

Model	**Attack**	**Defenses**
Image	Badnet[21], Clean-label[1]	De-trigger Autoencoder[29], IBAU[73], STRIP[20]
Audio	BAASV [75]	STRIP-VITA[19]
Text	BadNl[10]	ONION[51]
Graph	Zaixizhang[77], GTA[65]	N/A

Our contributions can be summarized as follows:

- We implemented Backdoor Pony, the first user-friendly web-based framework that allows smooth and fast evaluation of backdoor attacks and defenses.
- Backdoor Pony is the first tool that supports different applications like computer vision, text classification, audio recognition, and graph neural networks (GNNs).
- Like other tools, Backdoor Pony is extensible with arbitrary attacks and defenses in all four provided domains.

9.2. BACKDOOR ATTACKS ON NEURAL NETWORKS.

Recent trends like outsourced training, machine learning as a service (MLaaS), and crowdsourced datasets introduced a new threat for deep learning models, the backdoor attack [21]. In this attack, an adversary injects a secret functionality in a trained model that is activated through malicious inputs. In any other input, the model behaves normally to avoid raising any suspicions. This attack can be mounted through data poisoning [21], code poisoning [2], or weight poisoning [24]. Currently, we support only data poisoning as it is the most popular and represents the large majority of the related works.

In a classifier, this secret functionality may be targeted misclassifications that happens when the model's input contains the trigger. For example, a stop sign with a yellow post-it note (trigger) can be misclassified as a speed limit from a backdoored model deployed in an autonomous car [21]. In the recent literature, various attacks have been designed. The most targeted application is computer vision [18], but backdoor attacks have already been implemented against text [9, 49], audio [75, 27], and GNNs [65, 66].

Defending against backdoors is difficult as there is no method to exhaustively verify what a model has learned. New attacks that bypass existing countermeasures are continuously developed in the related literature, making this field a cat-and-mouse game between the attackers and the defenders [24]. There are various countermeasures already implemented that can be divided into different categories based on the assumptions used. Some of them remove the triggers from the inputs without knowing if the model is poisoned [16] or search for poisoned samples in the training dataset [5, 60]. Other inspect the trained model offline to reverse engineer a trigger pattern and unlearn the backdoor [62, 7], or monitor the operation of a model and raise alarms when there is suspicious behavior [20]. Some of these countermeasures are white-box and assume the defender has full access to the model and its training data [60, 5], or in the opposite case, black-box [20, 22]. This field is generally very active, and new attacks and defenses are published regularly, making it very challenging for researchers to keep up-to-date.

9.2.1. METRICS

We use the attack success rate (ASR) and the clean accuracy drop (CAD) to measure the backdoor's effectiveness. ASR is the percentage of the successfully activated backdoors over a number of tries and should be as high as possible. The clean accuracy drop shows the effect of the backdoor on the original task and should be as small as possible to avoid raising any suspicions. To see the effectiveness of a defense, we can either show the ASR's drop after the defense's implementation or a confusion matrix with statistics regarding poisoned and clean data samples. In this case, we have the following categories:

- **True/False positive rate**: the probability the classifier correctly/incorrectly identifies a poisoned input.
- **True/False negative rate**: the probability the classifier correctly/incorrectly identifies a benign input.

9.3. SOFTWARE DESCRIPTION

Backdoor Pony uses a client-server architecture, shown in *Figure 9.1*. The front-end consists of a web application with a GUI that controls the tool. The web application is

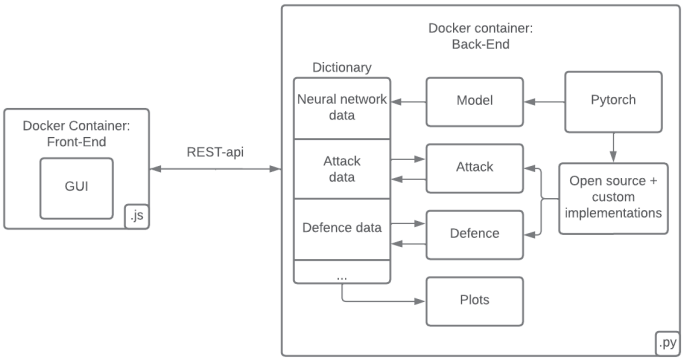


Figure 9.1: Software architecture overview.

written in javascript and mainly uses the Vue library. The back-end server is where all the services are run. The server is written in python and uses the Flask framework. Its `/src` can be found in the `server` folder, while the front-end `/src` can be found in the `gui` folder. The server and front-end use REST-API to communicate.

The server functions in parallel to the GUI interface. The highest level of the server's architecture consists of a runner that keeps track of the current state of the front-end application. Its job is to execute the instructions in the incoming REST requests on the appropriate service. The runner also ensures that the requests are made in the correct order, making it impossible, for example, to execute an attack if the model has not been chosen.

The different services executed in order are "model", "attack", "defense", and "plot". All the hyperparameters and the resulting data of the services are stored in a python dictionary used to pass information through the framework. The interaction between the services and the dictionary can be seen in *Figure 9.1*. After the user makes a selection from the available models and updates its hyperparameters, the model is trained on one of the datasets included in the framework. After the model is trained, the user selects and runs the attacks and defenses. Some of our implemented neural network models use PyTorch, and a few of our attacks and defenses also use the Adversarial Robustness Toolbox [44]. We implemented the rest of them from scratch using PyTorch. After applying the attacks and the defenses, we save all the corresponding hyperparameters and results into the dictionary. This information is used for the plots in the last step of our tool. The available metrics differ based on what attacks and defenses were selected. As stated before, the model service stores its results in the dictionary before the attacks and defenses are run. In this way, the user can easily compare the network's performance before or after the attack or the defense. Once the plot service has computed all the statistics, they are sent in a specific format to the GUI, which will generate the corresponding plots.

Backdoor Pony runs with Docker. The server and the front-end run in a separate container. Docker is also used to connect to the integrated GPU support. Running "docker-compose up" with an existing PyTorch image and CUDA/NVIDIA drivers will

automatically expose all available GPUs in the system to the container. The user may also select which specific GPUs will be used. One of our focuses for Backdoor Pony was to maintain high code quality. We have achieved this by implementing engineering practices, creating unit tests, and comprehensive documentation. This was done to ease future implementation or the integration of new attacks and defenses into Backdoor Pony.

9.4. ILLUSTRATIVE EXAMPLE

In this section, we first show a high-level overview of the framework's operating flow (*Figure 9.2*) and then provide an illustrative example of its usage. As we show in *Figure 9.2*, the user selects a dataset and a model, an attack (optionally), and a defense (optionally) and then proceeds to the training. When the training finishes, the user selects the plots needed and, optionally, downloads the execution history. Our tool has information buttons - the "*i*" letter enclosed in a red circular background - to facilitate its usage. When this button is clicked, a pop-up window shows all relevant information about that field.

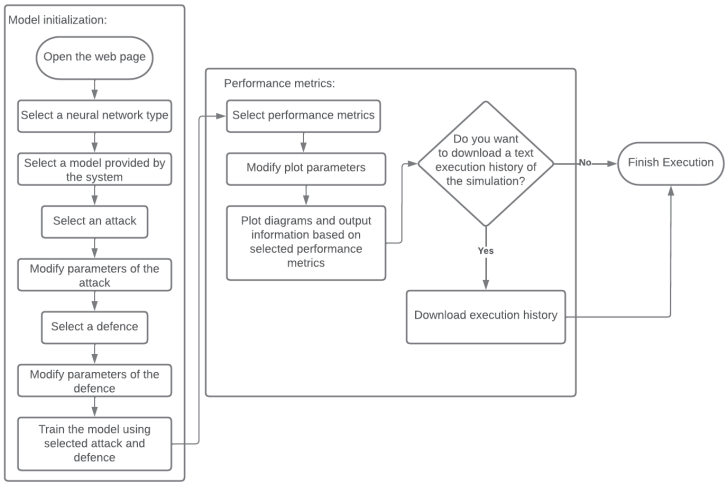


Figure 9.2: General workflow of the application.

9.4.1. DATASET SELECTION

First we need to select a dataset. The selection screen will appear after the user clicks on the **Start** button from the home page. Here, a set of available datasets is presented (*Figure 9.3*). Only one can be chosen at a time. Image, audio, text, and graph are the four types of available data categories, each having a unique icon. The right side of the screen displays a list of configurable hyperparameters for the model that will be trained on the selected dataset. In *Figure 9.3*, the user chooses the full CIFAR10 dataset (50000 training images), SGD as the model's optimizer, and a learning rate of 0.001.

When a choice has been made, the server loads a neural network that will be used to classify the data. If the framework has previously used this dataset, and the hyperparam-

eters were not changed from their default values, a pre-trained model will be loaded in the current execution. Otherwise, the network is trained with the data after clicking the **Continue** button.

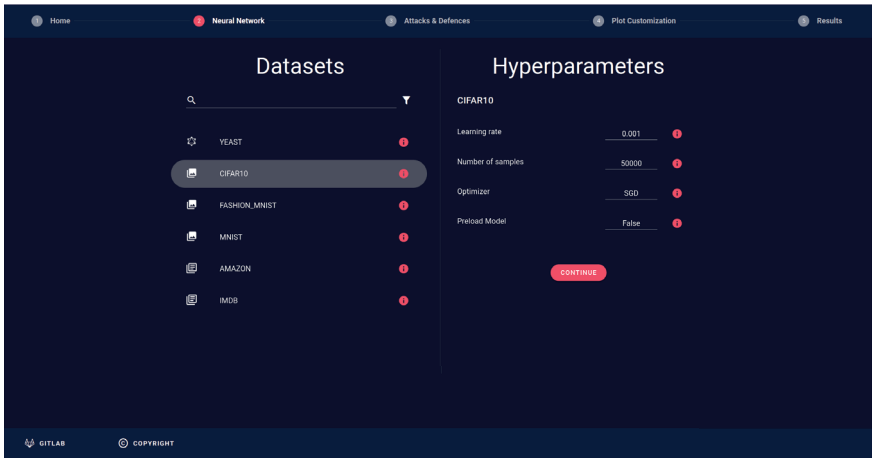


Figure 9.3: Dataset selection screen.

9.4.2. CHOOSING THE ATTACK AND DEFENSE

Next, the user selects an attack and a defense. Similarly to the datasets, attacks, and defenses are type-specific. Thus, only the attacks and defenses that match the dataset's type are shown for selection on the screen (*Figure 9.4*). Each attack and defense has a set of hyperparameters appearing on the right side of the page after selecting the desired mechanism (*Figure 9.4*). Multiple values can be typed in for a single field, separated by commas which will trigger an execution of the simulation for each unique combination of values.

Additionally, the user can skip running a defense for the current configuration by selecting the **None** label in the list of defenses. This choice impacts the available metrics after the attack has been completed. Once the user has chosen the means of attack, defense, and hyperparameters, clicking the **Execute** button - placed in the bottom right of the page (*Figure 9.4*) - will start applying the attack and defense on the neural network.

9.4.3. METRICS CONFIGURATION

The metrics' settings page will appear once the previous step has finished its execution. The tool can generate multiple plots that indicate the model's performance or the attack's and defense's effectiveness. These plots include the model's accuracy, the ASR, the CAD, the true and false positive rates, and the true and false negative rates.

To create a plot, the user clicks on the "**+**" sign of the desired metric and specifies the variables for the "***X***" and "***Y***" axes and the fixed hyperparameters selected for the attacks and defenses. An example plot would be the poisoned model's accuracy in the "***Y***" axis and the poisoning rate (the percentage of the poisoned data) in the "***X***" axis. The

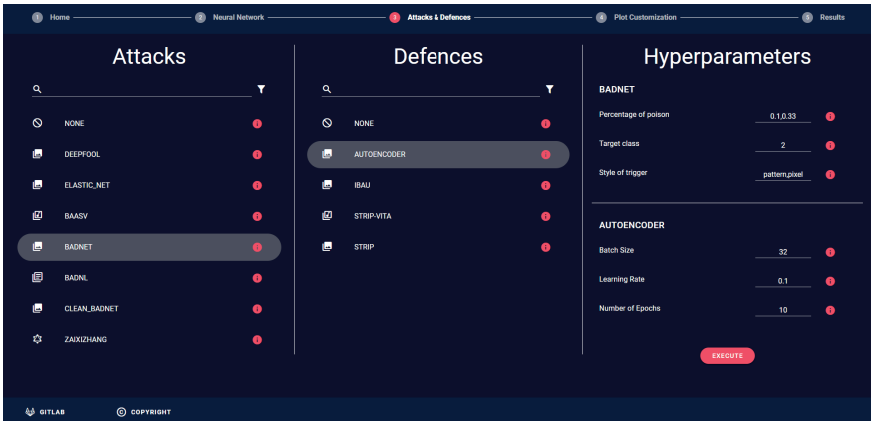


Figure 9.4: Attack (parameter selection upper right window) and Defense (parameters selection lower right window) selection screen.

fixed hyperparameters would be the rest of the options given by the user in the previous screens (target class, style of trigger) as shown in *Figure 9.5*. The **Add** button saves the plot configuration and the **Plot** button produces the corresponding plots. To create these plots, the server feeds clean or poisoned inputs to the model and calculates the chosen metrics.

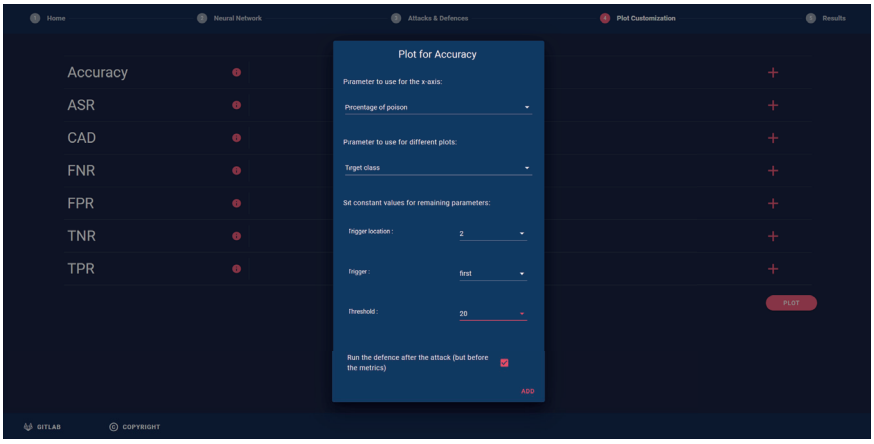


Figure 9.5: Network accuracy metric settings.

9.4.4. RESULTS

Finally, the plots will be presented in the **Results** tab (*Figure 9.6*), where we can see the clean accuracy (leftmost plot), which is the model's accuracy on clean inputs, the ASR (middle plot), which shows the attack's effectiveness, and the clean accuracy drop (rightmost plot), which shows the performance drop by the backdoor insertion. A text

file can be downloaded that includes details about the dataset, attacks, defenses, and the chosen hyperparameters. This file shows information about basic user choices in simple sentences and shows more complex options in a `json` format. An example of the generated configuration file is shown in Listing 9.1. It contains information about the chosen dataset, the model used, the attack and hyperparameters (trigger's position and poisoning rate), the defense and hyperparameters, and the plot configuration.



Figure 9.6: Example of resulting plots.

```
1. Select the IMDB dataset;
2. Select 'use built-in model';
3. Choose the stealthybadnl attack;
3a. Input [1] for target class;
  Attack parameters:
  {"target_class": {"pretty_name": "Target class", "value": [1]},
  "location": {"pretty_name": "Trigger location",  "value": ["start"],
       "values": ["start", "middle", "end"]},
  "trigger": {"pretty_name": "Trigger", "value": ["char"],
  "values": ["char", "word", "sentence"]},
  "poison_percent": {"pretty_name": "Percentage of poison", "value": [0.1],
            "minimum": 0, "maximum": 1, "isValid": true}}
4. Choose the onion defense;
Defense parameters:
{"threshold": {"pretty_name": "Threshold", "value": [20]}}
5. Press execute.
6. Metrics:
{"metrics0": {"metric": "Accuracy", "x_axis": "Percentage of poison",
              "plot": "Target class",
              "graph": {"1": {"points": [{"x": 0.1, "y": 50.8}],
                              "name": 1}}},
 "metrics1": {"metric": "Accuracy", "x_axis": "Percentage of poison",
              "plot": "Target class",
              "graph": {"1": {"points": [{"x": 0.1, "y": 52.8}],
                              "name": 1}}}}
```

Listing 9.1: Example configuration file

9.5. IMPACT

Backdoor Pony is a framework that targets both cutting-edge research and education. The framework's documentation shows how users can extend this tool through a few simple steps and add new attacks and defenses. This extension mechanism will keep Backdoor Pony up to date with all the recent developments in the field. An up-to-date tool will reduce the load of re-implementing existing works from the literature and lead to faster results and more accurate research outcomes. Additionally, Backdoor Pony can run multiple experiments sequentially by accepting a list of values for the experiment's hyperparameters. This speeds up ablation studies allowing users to focus on identifying trends instead of building pipelines. We have already used this tool in our internal research projects, and we believe it can be a valuable contribution to the community. Unlike current tools, e.g., [34, 63, 46], Backdoor Pony supports many application domains. Thus, it can become a common point of reference for experiments among researchers leading to fair comparisons between different attacks and defenses. Finally, its simplicity and user-friendliness will lower the barriers to entry in this field and attract a broader audience. As a result, it can become a valuable educational framework.

9.6. LIMITATIONS AND FUTURE WORK

While our tool can be a valuable contribution to the field we believe that there are a few limitations. First, a few popular attacks and defenses are missing. However, we plan to add them in the future allowing researchers to keep up with state-of-the-art without too much effort. Additionally, in this version, custom models and datasets are not supported as we wanted to ensure that our pipeline is functional before allowing arbitrary inputs from the users. In the future, we will add this feature to Backdoor Pony allowing the seamless evaluation of the attacks and the defenses on new data and tasks.

9.7. CONCLUSION

In this work, we presented Backdoor Pony which is a framework for evaluating backdoor attacks and defenses in various application domains. To our knowledge, Backdoor Pony is the first framework that supports four application domains (image, text, speech, and graphs) as most of the previous tools focus only on computer vision. It also offers full control of the hyperparameters involved through a user-friendly GUI making it suitable for cutting-edge research and education.

REFERENCES

[1] Aleksander Mądry Alexander Turner Dimitris Tsipras. *Clean-Label Backdoor Attacks*. URL: https://people.csail.mit.edu/madry/lab/cleanlabel.pdf.

[2] Eugene Bagdasaryan and Vitaly Shmatikov. "Blind Backdoors in Deep Learning Models". In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 1505–1521. ISBN: 978-1-939133-24-3. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/bagdasaryan.

[3] Mauro Barni, Kassem Kallas, and Benedetta Tondi. "A new backdoor attack in cnns by training set corruption without label poisoning". In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2019, pp. 101–105.

[4] Mariusz Bojarski et al. "Explaining how a deep neural network trained with end-to-end learning steers a car". In: *arXiv preprint arXiv:1704.07911* (2017).

[5] Bryant Chen et al. *Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering*. 2018. arXiv: 1811.03728 [cs.LG].

[6] Chuanshuai Chen and Jiazhu Dai. "Mitigating backdoor attacks in lstm-based text classification systems by backdoor keyword identification". In: *Neurocomputing* 452 (2021), pp. 253–262.

[7] Huili Chen et al. "DeepInspect: A Black-box Trojan Detection and Mitigation Framework for Deep Neural Networks". In: *IJCAI*. 2019.

[8] Sheng Chen and Hongxiang He. "Stock Prediction Using Convolutional Neural Network". In: *IOP Conference Series: Materials Science and Engineering* 435.1 (Oct. 2018), p. 012026. DOI: 10.1088/1757-899X/435/1/012026. URL: https://dx.doi.org/10.1088/1757-899X/435/1/012026.

[9] Xiaoyi Chen et al. "Badnl: Backdoor attacks against nlp models". In: *ICML 2021 Workshop on Adversarial Machine Learning*. 2021.

[10] Xiaoyi Chen et al. "BadNL: Backdoor Attacks against NLP Models with Semantic-preserving Improvements". In: *Annual Computer Security Applications Conference*. ACM, Dec. 2021. DOI: 10.1145/3485832.3485837. URL: https://doi.org/10.1145%2F3485832.3485837.

[11] Xinyun Chen et al. "Targeted backdoor attacks on deep learning systems using data poisoning". In: *arXiv preprint arXiv:1712.05526* (2017).

[12] Edward Chou, Florian Tramèr, and Giancarlo Pellegrino. *SentiNet: Detecting Localized Universal Attacks Against Deep Learning Systems*. 2018. arXiv: 1812.00292 [cs.CR].

[13] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. "Certified adversarial robustness via randomized smoothing". In: *international conference on machine learning*. PMLR. 2019, pp. 1310–1320.

[14] Ganqu Cui et al. "A Unified Evaluation of Textual Backdoor Learning: Frameworks and Benchmarks". In: *Proceedings of NeurIPS: Datasets and Benchmarks*. 2022.

[15] Jiazhu Dai, Chuanshuai Chen, and Yufeng Li. "A backdoor attack against lstm-based text classification systems". In: *IEEE Access* 7 (2019), pp. 138872–138878.

[16] Bao Gia Doan, Ehsan Abbasnejad, and Damith C. Ranasinghe. "Februus: Input Purification Defense Against Trojan Attacks on Deep Neural Network Systems". In: *Proceedings of the 36th Annual Computer Security Applications Conference (ACSAC)*. ACSAC 2020. Austin, TX, USA, 2020.

[17] Khoa Doan et al. "Lira: Learnable, imperceptible and robust backdoor attacks". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 11966–11976.

[18] Yansong Gao et al. “Backdoor attacks and countermeasures on deep learning: A comprehensive review”. In: *arXiv preprint arXiv:2007.10760* (2020).

[19] Yansong Gao et al. “Design and evaluation of a multi-domain Trojan detection method on deep neural networks”. In: *IEEE Transactions on Dependable and Secure Computing* (2021).

[20] Yansong Gao et al. “STRIP: A Defence against Trojan Attacks on Deep Neural Networks”. In: *Proceedings of the 35th Annual Computer Security Applications Conference*. ACSAC ’19. San Juan, Puerto Rico, USA: Association for Computing Machinery, 2019, pp. 113–125. ISBN: 9781450376280. DOI: 10.1145/3359789.3359790. URL: https://doi.org/10.1145/3359789.3359790.

[21] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. “Badnets: Identifying vulnerabilities in the machine learning model supply chain”. In: *arXiv preprint arXiv: 1708.06733* (2017).

[22] Wenbo Guo et al. *TABOR: A Highly Accurate Approach to Inspecting and Restoring Trojan Backdoors in AI Systems*. 2019. arXiv: 1908.01763 [cs.CR].

[23] Jonathan Hayase et al. “Spectre: Defending against backdoor attacks using robust statistics”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 4129–4139.

[24] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. “Handcrafted backdoors in deep neural networks”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 8068–8080.

[25] Kunzhe Huang et al. “Backdoor defense via decoupling the training process”. In: *arXiv preprint arXiv:2202.03423* (2022).

[26] Xijie Huang, Moustafa Alzantot, and Mani Srivastava. *NeuronInspect: Detecting Backdoors in Neural Networks via Output Explanations*. 2019. arXiv: 1911.07399 [cs.CR].

[27] Stefanos Koffas et al. “Can You Hear It? Backdoor Attacks via Ultrasonic Triggers”. In: *Proceedings of the 2022 ACM Workshop on Wireless Security and Machine Learning*. WiseML ’22. San Antonio, TX, USA: Association for Computing Machinery, 2022, pp. 57–62. ISBN: 9781450392778. DOI: 10.1145/3522783.3529523. URL: https://doi.org/10.1145/3522783.3529523.

[28] Keita Kurita, Paul Michel, and Graham Neubig. “Weight poisoning attacks on pre-trained models”. In: *arXiv preprint arXiv:2004.06660* (2020).

[29] Hyun Kwon. “Defending Deep Neural Networks against Backdoor Attack by Using De-trigger Autoencoder”. In: *IEEE Access* (2021), pp. 1–1. DOI: 10.1109/ACCESS.2021.3086529.

[30] Linyang Li et al. “Backdoor attacks on pre-trained models by layerwise weight poisoning”. In: *arXiv preprint arXiv:2108.13888* (2021).

[31] Yige Li et al. “Anti-backdoor learning: Training clean models on poisoned data”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 14900–14912.

[32] Yige Li et al. “Neural attention distillation: Erasing backdoor triggers from deep neural networks”. In: *arXiv preprint arXiv:2101.05930* (2021).

[33] Yiming Li et al. “Backdoor attack in the physical world”. In: *arXiv preprint arXiv:2104.02361* (2021).

[34] Yiming Li et al. “BackdoorBox: A Python Toolbox for Backdoor Learning”. In: (2022).

[35] Yuezun Li et al. “Invisible backdoor attack with sample-specific triggers”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 16463–16472.

[36] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. “Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks”. In: *Research in Attacks, Intrusions, and Defenses*. Springer International Publishing, 2018, pp. 273–294. ISBN: 9783030004705. DOI: 10.1007/978-3-030-00470-5_13. URL: http://dx.doi.org/10.1007/978-3-030-00470-5_13.

[37] Yingqi Liu et al. “Trojaning Attack on Neural Networks”. In: *25nd Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-221, 2018*. The Internet Society, 2018.

[38] Yunfei Liu et al. “Reflection backdoor: A natural backdoor attack on deep neural networks”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*. Springer. 2020, pp. 182–199.

[39] Yuntao Liu, Yang Xie, and Ankur Srivastava. “Neural trojans”. In: *2017 IEEE International Conference on Computer Design (ICCD)*. IEEE. 2017, pp. 45–48.

[40] Aleksander Madry et al. “Towards deep learning models resistant to adversarial attacks”. In: *arXiv preprint arXiv:1706.06083* (2017).

[41] Dongyu Meng and Hao Chen. “Magnet: a two-pronged defense against adversarial examples”. In: *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 2017, pp. 135–147.

[42] Anh Nguyen and Anh Tran. “Wanet–imperceptible warping-based backdoor attack”. In: *arXiv preprint arXiv:2102.10369* (2021).

[43] Tuan Anh Nguyen and Anh Tran. “Input-aware dynamic backdoor attack”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 3454–3464.

[44] Maria-Irina Nicolae et al. “Adversarial Robustness Toolbox v1. 0.0”. In: *arXiv preprint arXiv:1807.01069* (2018).

[45] Ren Pang et al. “A tale of evil twins: Adversarial inputs versus poisoned models”. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 2020, pp. 85–99.

[46] Ren Pang et al. “TrojanZoo: Towards Unified, Holistic, and Practical Evaluation of Neural Backdoors”. In: *Proceedings of IEEE European Symposium on Security and Privacy (Euro S&P)*. 2022.

[47] P Jonathon Phillips and Alice J O'toole. "Comparison of human and computer performance across face recognition experiments". In: *Image and Vision Computing* 32.1 (2014), pp. 74–85.

[48] Fanchao Qi et al. "Hidden killer: Invisible textual backdoor attacks with syntactic trigger". In: *arXiv preprint arXiv:2105.12400* (2021).

[49] Fanchao Qi et al. "Mind the Style of Text! Adversarial and Backdoor Attacks Based on Text Style Transfer". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2021, pp. 4569–4580.

[50] Fanchao Qi et al. "Mind the style of text! adversarial and backdoor attacks based on text style transfer". In: *arXiv preprint arXiv:2110.07139* (2021).

[51] Fanchao Qi et al. "Onion: A simple and effective defense against textual backdoor attacks". In: *arXiv preprint arXiv:2011.10369* (2020).

[52] Fanchao Qi et al. "Turn the combination lock: Learnable textual backdoor attacks via word substitution". In: *arXiv preprint arXiv:2106.06361* (2021).

[53] Xiangyu Qi et al. "Revisiting the assumption of latent separability for backdoor defenses". In: *The Eleventh International Conference on Learning Representations*. 2023.

[54] Lujia Shen et al. "Backdoor pre-trained models can transfer to all". In: *arXiv preprint arXiv:2111.00197* (2021).

[55] Reza Shokri et al. "Bypassing backdoor detection algorithms in deep learning". In: *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2020, pp. 175–183.

[56] Hossein Souri et al. "Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch". In: *arXiv preprint arXiv:2106.08970* (2021).

[57] Di Tang et al. "Demon in the variant: Statistical analysis of dnns for robust backdoor contamination detection". In: *30th {USENIX} Security Symposium ({USENIX} Security 21)*. 2021.

[58] Ruixiang Tang et al. "An embarrassingly simple approach for trojan attack in deep neural networks". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 218–228.

[59] Xie Tinghao. *Backdoor Toolbox*. [Online, Accessed: 31 March 2023]. 2022. URL: https://github.com/vtu81/backdoor-toolbox.

[60] Brandon Tran, Jerry Li, and Aleksander Mądry. "Spectral Signatures in Backdoor Attacks". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Montréal, Canada: Curran Associates Inc., 2018, pp. 8011–8021.

[61] Sakshi Udeshi et al. "Model agnostic defence against backdoor attacks in machine learning". In: *IEEE Transactions on Reliability* 71.2 (2022), pp. 880–895.

[62] B. Wang et al. "Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks". In: *2019 IEEE Symposium on Security and Privacy (SP)*. 2019, pp. 707–723.

[63] Baoyuan Wu et al. “BackdoorBench: A Comprehensive Benchmark of Backdoor Learning”. In: 2022.

[64] Dongxian Wu and Yisen Wang. “Adversarial neuron pruning purifies backdoored deep models”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 16913–16925.

[65] Zhaohan Xi et al. “Graph backdoor”. In: *30th USENIX Security Symposium (USENIX Security 21)*. 2021, pp. 1523–1540.

[66] Jing Xu et al. “More is better (mostly): On the backdoor attacks in federated graph neural networks”. In: *arXiv preprint arXiv:2202.03195* (2022).

[67] Weilin Xu, David Evans, and Yanjun Qi. “Feature squeezing: Detecting adversarial examples in deep neural networks”. In: *arXiv preprint arXiv:1704.01155* (2017).

[68] Wenkai Yang et al. “Be careful about poisoned word embeddings: Exploring the vulnerability of the embedding layers in nlp models”. In: *arXiv preprint arXiv:2103.15543* (2021).

[69] Wenkai Yang et al. “Rap: Robustness-aware perturbations for defending against backdoor attacks on nlp models”. In: *arXiv preprint arXiv:2110.07831* (2021).

[70] Wenkai Yang et al. “Rethinking stealthiness of backdoor attack against nlp models”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2021, pp. 5543–5557.

[71] Yang Yang et al. *TI-CNN: Convolutional Neural Networks for Fake News Detection*. 2018. DOI: 10.48550/ARXIV.1806.00749. URL: https://arxiv.org/abs/1806.00749.

[72] Yuanshun Yao et al. “Latent backdoor attacks on deep neural networks”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019, pp. 2041–2055.

[73] Yi Zeng et al. *Adversarial Unlearning of Backdoors via Implicit Hypergradient*. 2021. DOI: 10.48550/ARXIV.2110.03735. URL: https://arxiv.org/abs/2110.03735.

[74] Yi Zeng et al. “Rethinking the backdoor attacks’ triggers: A frequency perspective”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 16473–16481.

[75] Tongqing Zhai et al. “Backdoor attack against speaker verification”. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 2560–2564.

[76] Xinyang Zhang et al. “Trojaning language models for fun and profit”. In: *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2021, pp. 179–197.

[77] Zaixi Zhang et al. *Backdoor Attacks to Graph Neural Networks*. 2020. DOI: 10.48550/ARXIV.2006.11165. URL: https://arxiv.org/abs/2006.11165.

[78] Zhengyan Zhang et al. “Red alarm for pre-trained models: Universal vulnerability to neuron-level backdoor attacks”. In: *Machine Intelligence Research* (2023), pp. 1–14.

[79] Pu Zhao et al. “Bridging mode connectivity in loss landscapes and adversarial robustness”. In: *arXiv preprint arXiv:2005.00060* (2020).

[80] Shihao Zhao et al. “Clean-label backdoor attacks on video recognition models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 14443–14452.

IV

Backdoor Attacks and Defenses in Depth

10

BAN: Detecting Backdoors Activated by Adversarial Neuron Noise

Backdoor attacks on deep learning represent a recent threat that has gained significant attention in the research community. Backdoor defenses are mainly based on backdoor inversion, which has been shown to be generic, model-agnostic, and applicable to practical threat scenarios. State-of-the-art backdoor inversion recovers a mask in the feature space to locate prominent backdoor features, where benign and backdoor features can be disentangled. However, it suffers from high computational overhead, and we also find that it overly relies on prominent backdoor features that are highly distinguishable from benign features. To tackle these shortcomings, this paper improves backdoor feature inversion for backdoor detection by incorporating extra neuron activation information. In particular, we adversarially increase the loss of backdoored models with respect to weights to activate the backdoor effect, based on which we can easily differentiate backdoored and clean models. Experimental results demonstrate our defense, BAN, is 1.37× (on CIFAR-10) and 5.11× (on ImageNet200) more efficient with an average 9.99% higher detect success rate than the state-of-the-art defense BTI-DBF. Our code and trained models are publicly available at https://github.com/xiaoyunxxy/ban.

This chapter has been published as X. Xu, Z. Liu, S. Koffas, S. Yu, S. Picek, (2024). "BAN: detecting backdoors activated by adversarial neuron noise." *Advances in Neural Information Processing Systems, 37, 114348-114373.*

10.1. INTRODUCTION

Deep neural networks (DNNs) are known to be vulnerable to backdoor attacks, a setting where the attacker trains a malicious DNN to perform well on normal inputs but behave inconsistently when receiving malicious inputs that are stamped with triggers [9]. The malicious model is obtained by training with poisoned data [9, 6], by tampering with the training process [22, 2], or by directly altering the model's weights [14]. Backdoors are proposed for various domains, from computer vision [9, 22, 6, 26, 38, 25] to graph data [42] and neuromorphic data [1]. Still, backdoors in computer vision received most of the attention of the research community, which also means there is a significant variety of triggers. For instance, a trigger can be a pixel patch [9], an existing image [22], dynamic perturbation [25], or image warping [26]. Various backdoor attacks target different stages of the machine learning model pipeline, inducing several threat scenarios in which defenses are developed by exploiting different knowledge.

Trigger inversion is a principled method that makes minimal assumptions about the backdoor attacks in the threat model [39, 40, 44], and it has clear advantages to training-time [18] or run-time defenses [24, 19, 11]. *Input space* trigger inversion is first introduced by Neural Cleanse (NC) [36], where potential triggers for all target classes are reversed by minimizing the model loss of clean inputs. Median absolute deviations of reversed triggers from all classes are then calculated to detect triggers. More recently, input space trigger inversion methods have been shown to be ineffective against *feature space* backdoor attacks [26, 39]. To address this problem, FeatureRE [39] proposes a detection method using feature space triggers. The authors observe that features of backdoored and benign inputs are separable in the feature space by hyperplanes. Unicorn [40] formally defines and analyzes different spaces for trigger inversion problems. An invertible input space transformation function is proposed to map the input space to others, such as feature space, and to reconstruct the backdoor triggers. These trigger inversion methods can be considered as an optimization problem for the targeted class. They optimize the input images to mislead the model under various constraints and recover strong triggers in different spaces. The state-of-the-art trigger inversion method, BTI-DBF [44], increases the inversion efficiency by relaxing the dependency of trigger inversion on the target labels. BTI-DBF trains a trigger generator by minimizing the differences between benign samples and their generated poisoned version in decoupled benign features while maximizing the differences in remaining backdoor features. Feature space trigger inversion defenses are generic and effective against most backdoor attacks. However, we show that BTI-DBF may fail against BadNets, as its backdoor is not prominent in the feature space (see Section 10.3.4). In addition, existing works still suffer from huge computational overhead and overly rely on *prominent backdoor features*.

To resolve this shortcoming, we propose a defense called detecting **B**ackdoors activated by **A**dversarial neuron **N**oise (BAN). Our defense is inspired by the finding that backdoored models are more sensitive to adversarial noise than benign models [43, 10], and neuron noise can be adversarially manipulated to indicate backdoor activations [41]. Specifically, BAN generates adversarial neuron noise, where the weights of the victim model are adversarially perturbed to maximize the classification loss on a clean data set. Simultaneously, trigger inversion is conducted in the victim model's feature space to calculate the mask for benign and backdoor features. Clean inputs with masked feature

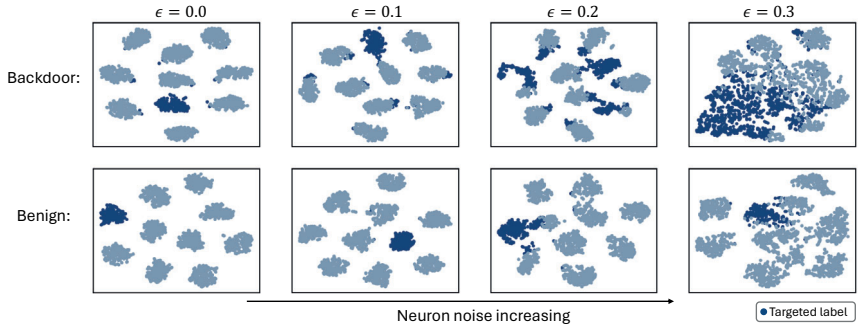


Figure 10.1: The feature plots of backdoor and benign models with neuron noise using ResNet18 on CIFAR-10. The darker blue represents the target label. As noise increases, the backdoor model identifies more inputs from each class as the target label. The clean model has fewer errors, and there is no significant increase in the number of misclassifications to the target class.

maps are then fed to the adversarially perturbed model, based on the outputs of which backdoored models can be differentiated. Figure 10.1 presents a t-SNE visualization of the feature space for backdoored and clean models when they are perturbed with different levels of adversarial neuron noise. It can be observed that the backdoored model misclassifies parts of the data from all classes as the target class when the adversarial neuron noise increases, while the clean model has fewer misclassifications under the same level of noise. By leveraging the induced difference, BAN can successfully detect and mitigate backdoor attacks in both input and feature space.

We make the following contributions:

- We find a generalization shortcoming in current trigger inversion-based detection methods (FeatureRE [39], Unicorn [40], BTI-DBF [44]). In particular, feature space detections overly rely on highly distinguishable prominent backdoor features. We provide an in-depth analysis of trigger inversion-based backdoor defenses, showing that prominent backdoor features that are exploited by state-of-the-art defenses to distinguish feature space backdoors may not be suitable for the identification of input space backdoors.
- We propose detecting Backdoors activated by Adversarial neuron Noise (BAN) to mitigate this generalization shortcoming by introducing neuron noise into feature space trigger inversion. BAN includes an adversarial learning process to incorporate neuron activation information into the inversion-based backdoor detection. Experimental results demonstrate that BAN is 1.37× (on CIFAR-10) and 5.11× (on ImageNet200) more efficient with a 9.99% higher detect success rate than the state-of-the-art defense BTI-DBF [44].
- We also exploit the neuron noise to further design a simple yet effective defense for removing the backdoor, such that we build a workable framework.

10.2. BACKDOORS

Attacks. Backdoor attacks [9, 22, 2, 26, 25, 38, 35, 8, 20, 3] refer to injecting a secret functionality into the victim model that is activated through malicious inputs that contain the trigger. To this end, substantial research has been proposed by poisoning a small percentage of training data using small and static triggers [9, 6, 22]. Early attacks generate backdoors in input space, where BadNets [9] is the first backdoor attack in DNNs. Blend [6] proposed three injection strategies to blend translucent images into inputs of DNNs as triggers. The authors controlled the transparency of the trigger to allow the trade-off between strength of attack and invisibility. Although these attacks work well, their triggers are still perceptible to humans and can be easily detected by backdoor defenses, such as Activation Clustering (AC) [4] and NC [36].

Dynamic and imperceptible triggers [26, 25, 38, 8], including feature space backdoor triggers [20, 25], are explored to bypass both human observers and input space defenses. IAD [25] designs input-specific triggers. To evaluate the uniqueness of dynamic triggers, the authors designed a cross-trigger test to determine whether the trigger of one input is reusable to others. WaNet [26] proposes warping-based triggers, which are unnoticeable and smooth in the input space. Bpp [38] exploits vulnerabilities in the human visual system and, by using image quantization and dithering, introduces invisible triggers that are stealthier than previous attacks.

Adaptive backdoor attacks [31, 28, 24] are built to systematically evaluate defenses, where attacks discourage the indistinguishability of latent representations of poisoned and benign inputs in the feature space. Adap-Blend [28] divides the trigger image into 16 pieces and randomly applies only 50% of the trigger during data poisoning. They use the full trigger image at inference time to mislead the poisoned model. SSDT [24] considers the combination of source-specific and dynamic-trigger backdoors. Only the inputs from victim classes (source) with the dynamic triggers are classified to the target labels, which encourages the generalization of more diverse trigger patterns from different inputs. In this paper, we evaluate BAN against various types of attacks, including input space, feature space, and adaptive attacks, to provide a systematic evaluation resembling practical threats.

Defenses. Trigger inversion-based backdoor defense is considered one of the most practical and general defenses against backdoors [39, 40, 44]. The recovered trigger is used to determine whether the model is backdoored. For example, NC [36] reverses input space triggers to detect backdoors by selecting significantly smaller triggers in size. Other methods, such as ABS [21] and FeatureRE [39], usually determine whether there is a backdoor based on the attack success rate of the trigger. More specifically, given a DNN model f and a small set of clean data $\mathscr{D}_c = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, NC recovers the potential trigger by solving the following objective:

$$\min_{\mathbf{m},\mathbf{t}} \mathscr{L}(f((1-\mathbf{m}) \odot \mathbf{x} + \mathbf{m} \odot \mathbf{t}), y_t) + \lambda |\mathbf{m}|, \tag{10.1}$$

where $(\mathbf{x}, y) \in \mathscr{D}_c$ and $\mathbf{m}$ is the trigger mask, $\mathbf{t}$ is the trigger pattern, and y_t is the target label. The mask $\mathbf{m}$ determines whether the pattern will replace the pixel. $\mathscr{L}$ is the cross-entropy loss function. Most prior works [12, 30, 21, 39, 40] follow this design to conduct trigger inversion for all possible target labels. For example, Tabor [12] adds more constraints to the NC optimization problem according to the overly large (large size triggers but no

overlaps with the true trigger) and overlaying (overlap the true trigger but with redundant noise) problem of NC. Moving from input space to feature, FeatureRE [39] utilizes feature space constraint according to an observation that neuron activation values representing the backdoor behavior are orthogonal to others. Unicorn [40] proposes a transformation function that projects from the input space to other spaces, such as feature space [26] and numerical space [38]. Then, the authors conduct trigger inversion in different spaces.

Unlike previous NC-style methods, recent works [44, 37] explore different optimization objectives to avoid the time-consuming optimization for all possible target classes or to avoid the fixed mask-pattern design as advanced attacks utilize more complex and dynamic triggers. BTI-DBF [44] takes advantage of the prior knowledge that benign and backdoored features are decoupled in the feature space. It distinguishes them by the optimization objective, where benign features contribute to the correct predictions and backdoored features lead to wrong predictions. Based on the decoupled features, BTI-DBF trains a trigger generator by minimizing the difference between benign samples and their generated version according to the benign features and maximizing the difference according to the backdoored features. Feature space backdoor defenses are developed based on the fact that backdoor features are highly distinguishable from benign features. However, this finding is not consistently valid for input space attacks where the feature difference is small (See Section 10.3.4).

10.3. BAN METHOD

10.3.1. THE PIPELINE OF TRAINING BACKDOOR MODELS

For brevity, consider an L-layer fully connected network f (similar principles apply to convolutional networks) that has $l = l_1 + l_2 + \cdots + l_L$ neurons. $\mathbf{x}_n \in \mathbb{R}^{d_x}$ and $y_n \in \{0,1\}^{d_y}$ are the n_{th} image and its label in d_x and d_y dimensional spaces, respectively. The attacker creates a poisoned dataset $\mathscr{D}_p$ by poisoning generators G_X and G_Y for a subset of the whole training dataset, i.e., $\mathscr{D}_p = \mathscr{D}_c \cup \mathscr{D}_b$. $\mathscr{D}_c$ is the original clean data. $\mathscr{D}_b$ is the poisoned backdoor data, $\mathscr{D}_b = \{(\mathbf{x}', y') | \mathbf{x}' = G_X(\mathbf{x}), y' = G_Y(y), (\mathbf{x}, y) \in \mathscr{D} - \mathscr{D}_c\}$. In all-to-one attacks, $G_Y(y) = y_t$, y_t is the attacker-specified target class. In our experiments, we consider the dirty-label attack. In all-to-all attacks, ususally $G_Y(y) = (y+1)$ [44, 26, 38], which is also what we chose in our experiments. In the training stage, the backdoor is injected into the model by training with $\mathscr{D}_p$, i.e., minimizing the training loss on $\mathscr{D}_p$ to find the optimal weights and bias $(\mathbf{w}^*, \mathbf{b}^*)$:

$$\min_{\mathbf{w},\mathbf{b}} \mathscr{L}_{\mathscr{D}_p}(\mathbf{w},\mathbf{b}) = \mathop{\mathbb{E}}_{(\mathbf{x},y)\in\mathscr{D}_p} \ell(f(\mathbf{x};\mathbf{w},\mathbf{b}), y), \tag{10.2}$$

where $\ell(\cdot,\cdot)$ is the cross-entropy. In the inference stage, the backdoored model predicts an unseen input $\hat{\mathbf{x}}$ as its true label $\hat{y}$ but predicts $G_X(\hat{\mathbf{x}})$ as $G_Y(\hat{y})$: $f(G_X(\hat{\mathbf{x}});\mathbf{w},\mathbf{b}) = G_Y(\hat{y})$.

10.3.2. THREAT MODEL

Attacker's goal. We consider the attacker to be the pre-trained model's provider. The attacker aims to inject stealthy backdoors into the pre-trained models. The pre-trained models perform well on clean inputs but predict the attacker-chosen target label when receiving backdoor inputs.

Attacker knowledge. The attacker has white-box access to the model, including training data, architectures, hyperparameters, and model weights.

Defender's goal and knowledge. The main goal is to detect whether a given model is backdoored and then remove the potential backdoor according to the detection results. Following [39, 40, 44], we assume the defender has white-box access to the model and holds a few local clean samples. However, the defender does not have access to the training data and has no knowledge of the backdoor trigger.

10.3.3. DETECTION WITH NEURON NOISE

Based on previous findings that adversarial noise can activate backdoors [43, 41, 10], we design a two-step method for backdoor detection. First, we search for noise on neurons that can activate the potential backdoor. Then, we decouple the benign and backdoored features using a learnable mask of the latent feature (the output before the final classification layer).

Neuron Noise. In backdoor models, there are two types of neurons: benign and backdoor [21]. The backdoor neurons build a shortcut from $G_X(\mathbf{x})$ to $G_Y(y)$. *Neuron noise* is generated noises added on neurons to maximize model loss on clean samples in an adversarial manner [23]. The noise on benign neurons evenly misleads prediction to all classes, while the noise on backdoor neurons tends to mislead prediction to the target label due to the backdoor shortcut, as shown in Figure 10.1. Therefore, backdoor models with noise behave differently from benign models with noise, as there are no backdoor neurons in the benign models.

Given the j_{th} neuron connecting the i_{th} layer and $(i-1)_{\text{th}}$ layer, we denote its weight and bias with $\mathbf{w}_{ij}$ and b_{ij}, respectively. Neuron noise can be added to the neuron by multiplying its weight and bias with a small number: $(1+\delta_{ij})\mathbf{w}_{ij}$ and $(1+\xi_{ij})b_{ij}$. Then, the output of the neuron is:

$$h_{ij} = \sigma\big((1+\delta_{ij})\mathbf{w}_{ij}^{\top}\mathbf{h}_{(i-1)} + (1+\xi_{ij})b_{ij}\big), \tag{10.3}$$

where $\mathbf{h}_{(i-1)}$ is the output of the previous layer and $\sigma(\cdot)$ is the nonlinear function (activation). The noise on all neurons are represented by $\boldsymbol{\delta} = [\delta_{1,1}, \cdots, \delta_{l_1,1}, \cdots, \delta_{1,L}, \cdots, \delta_{l_L,L}]$ and $\boldsymbol{\xi} = [\xi_{1,1}, \cdots, \xi_{l_1,1}, \cdots, \xi_{1,L}, \cdots, \xi_{l_L,L}]$. The $\boldsymbol{\delta}$ and $\boldsymbol{\xi}$ are optimized via a maximization to increase the cross-entropy loss on the clean data:

$$\begin{aligned} &\max_{\boldsymbol{\delta},\boldsymbol{\xi}\in S} \mathscr{L}_{\mathscr{D}_c}((1+\boldsymbol{\delta})\odot\mathbf{w}, (1+\boldsymbol{\xi})\odot\mathbf{b}), \\ &S = B(\mathbf{w};\epsilon) = \{\delta \in \mathbb{R}^l | \delta \le \epsilon\}, \end{aligned} \tag{10.4}$$

where S is the ball function of the radius ϵ in the l dimensional space. $\boldsymbol{\delta}$ and $\boldsymbol{\xi}$ share the ball function S as the maximum noise size. The maximization in Eq. (10.4) can be solved by PGD [23] algorithm with a random start, as PGD can better explore the entire searching space to mitigate local minima [23].

Feature Decoupling with Mask. The neuron noise activates the backdoor (see Figure 10.1) and misleads the predictions to the target label. However, the performance has high variance when searching for noise multiple times, which we conjecture is caused by random initialization. Therefore, inspired by [44], we further introduce a feature decoupling process to enhance the effect of noise on backdoored features but maintain

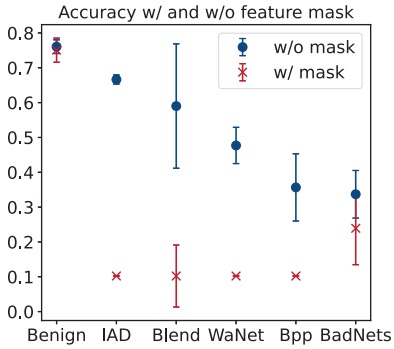


Figure 10.2: Model's clean accuracy with (red dots) and without (blue dots) the mask defined in Eq. (10.6). Only the backdoored models are affected by the noise.

a decreased effect on benign features. Specifically, the network f is decomposed into $g = f_1 \circ \cdots \circ f_{L-1}$ and f_L, where f_{L-1} extracts the latent features from $\mathbf{x}$, while f_L is the classification layer. Then, we use a mask $\mathbf{m}$ on top of the latent feature $g(\mathbf{x})$ to decouple benign and backdoored features. The optimization objective can be written as:

$$\min_{\mathbf{m}} \mathscr{L}(f_L \circ (g(\mathbf{x}) \odot \mathbf{m}), y) - \mathscr{L}(f_L \circ (g(\mathbf{x}) \odot (1-\mathbf{m})), y) + \lambda_1 |\mathbf{m}|, \tag{10.5}$$

where $\odot$ is the element-wise product. This optimization divides the latent features into two parts through the mask while maintaining a relatively small size of $\mathbf{m}$. Note that the regularizer for $\mathbf{m}$ in Eq. (10.5) is necessary. Otherwise, $\mathbf{m}$ will become a dense matrix full of ones to focus on the positive part of Eq. (10.5) because maintaining only the positive part (without penalty of $|\mathbf{m}|$) already satisfies the optimization objective. Finally, we apply the negative mask $(1-\mathbf{m})$ on top of latent feature of f to enhance the backdoor effect. The final output is:

$$f_L \circ \Big(g\big(\mathbf{x}; (1+\boldsymbol{\delta}) \odot \mathbf{w}, (1+\boldsymbol{\xi}) \odot \mathbf{b}\big) \odot (1-\mathbf{m}) \Big). \tag{10.6}$$

In Figure 10.2, the blue dots show the accuracy after adding noise to the model. The red dots show the accuracy (with noise) while applying the feature mask to the model using Eq. (10.6). Our feature masks do not affect the performance of benign models. However, we see that the performance is significantly decreased for backdoored models. This decrease is caused by the model only using backdoor features (through the negative mask), which means the backdoor is activated more frequently. Finally, a suspect model is determined backdoored if the prediction using Eq. (10.6) has a high attack success rate. For all-to-one attacks, the misclassification will be concentrated into one label, which is the target label. For all-to-all attacks, we can do the same as for all-to-one attacks but evaluate the prediction for each class independently.

10.3.4. IMPROVING BTI-DBF

This section shows that the most recent work, BTI-DBF [44], may fail to capture the backdoor features by its decoupling method in Table 10.2. We show how to patch BTI-DBF using a simple solution. The BTI-DBF is the original version, and BTI-DBF* is our

improved version. The main pipeline of BTI-DBF consists of two steps: (1) decoupling benign and backdoor features and (2) trigger inversion by minimizing the distance between benign features and maximizing the distance between backdoor features. We found that the defense's first step may introduce errors in the decoupled features. Similar to our Eq. (10.5), the decoupling of BTI-DBF can be written as:

$$\min_{\mathbf{m}} \mathscr{L}(f_L \circ (g(\mathbf{x}) \odot \mathbf{m}), y) - \mathscr{L}(f_L \circ (g(\mathbf{x}) \odot (1-\mathbf{m})), y). \tag{10.7}$$

However, this equation has no constraints on the feature mask $\mathbf{m}$. Overall, the optimization objective is to decrease the loss. Obviously, BTI-DBF's decoupling encourages the norm of the mask to increase so that the loss will focus on the positive part and ignore the negative part because the negative part goes against the overall objective. Finally, the mask will be a dense matrix that is full of ones. The $f_L \circ (g(\mathbf{x}) \odot (1-\mathbf{m}))$ is ignored due to multiplying by zero. We propose a simple solution to fix the problem by adding a regularizer of the size of the mask to the loss, i.e., we use our Eq. (10.5) as the first step of BTI-DBF*. According to Table 10.2, BTI-DBF* successfully overcomes BTI-DBF's shortcomings.

10.3.5. BACKDOOR DEFENSE

After determining whether a suspect model is backdoored, we can fine-tune the backdoored model to remove the backdoor. However, standard fine-tuning using clean data does not effectively remove the backdoor because it does not activate it. Therefore, we propose using optimized neuron noise to fine-tune the model. In the optimization of the neuron noise, the objective is to increase the loss of $\mathscr{L}(f(\mathbf{x}), y)$. We consider both the benign and backdoor neurons to contribute to the increase in loss when optimizing the noise. Indeed, on benign neurons, the neuron noise misleads f to any result other than the true label y, while the noise misleads f to the target label $G_Y(y)$ on backdoor neurons. Therefore, a straightforward method is to decrease the loss between noise output and the true label. The loss for our noise fine-tuning can be written as:

$$\min_{\mathbf{w},\mathbf{b}} \mathscr{L}(f(\mathbf{x};\mathbf{w},\mathbf{b}), y) + \lambda_2 \mathscr{L}(f(\mathbf{x};(1+\boldsymbol{\delta})\odot\mathbf{w},(1+\boldsymbol{\xi})\odot\mathbf{b}), y). \tag{10.8}$$

10.4. EXPERIMENTAL SETUP

10.4.1. DATASETS

The datasets for our experiments include CIFAR-10 [16], GTSRB [33], Tiny-ImageNet [17], and a subset of ImageNet-1K [7]. **CIFAR-10** [16] contains 50,000 training images and 10,000 testing images with the size of $3 \times 32 \times 32$ in 10 classes. **GTSRB** [33] contains 39,209 training images and 12,630 testing images in 43 classes. In our experiments, the images are resized to $3 \times 32 \times 32$. **Tiny-ImageNet** [17] contains 100,000 training images and 10,000 testing images with the size $3 \times 64 \times 64$ in 200 classes. **ImageNet** [7] contains over 1.2 million high-resolution images in 1,000 classes. Our experiments use a subset of the full ImageNet dataset, i.e., 200 randomly selected classes. Each class has 1300 training images and 50 testing images. The ImageNet images are scaled to $3 \times 224 \times 224$.

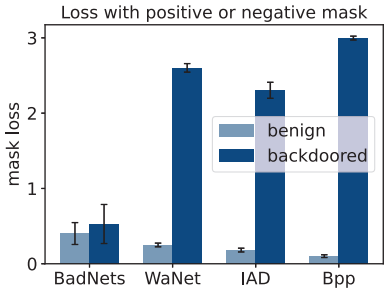


Figure 10.3: BadNets features are weaker when using the mask to disentangle the benign and backdoor features. Defenses that are biased towards large differences may not work in cases like BadNets.

10.4.2. MODELS

BAN is evaluated using three architectures: ResNet18 [13], VGG16 [32], and DenseNet121 [15]. For all backdoored models, we use SGD with a momentum of 0.9, weight decay of 5×10^{-4}, and a learning rate of 0.01. We train for 200 epochs, and the learning rate is divided by 10 at the 100-th and 150-th epochs. Same to [41], we use 90% of training data for training backdoor models and 10% for the validation set.

10.4.3. ATTACK BASELINES.

Our experiments are conducted using seven attacks: BadNets [9], Blend [6], WaNet [26], IAD [25], BppAttack [38], Adap-Blend [28], and SSDT [24], which are commonly used in other works [44, 39, 40, 46, 29]. The BadNets [9] and Blend [6] are designed for input space. WaNet [26], IAD [25], and BppAttack [38] are designed for feature space. Adap-Blend [28] and SSDT [24] (for adaptive evaluation) are state-of-the-art attacks that have been recently introduced to bypass backdoor defenses. The main idea of Adap-Blend [28] and SSDT [24] is to obscure the latent separation in features of benign and backdoor samples. We used the following configuration parameters for our triggers:

- **BadNets.** We use a 3×3 pattern to build the trigger with 5% poisoning rate.
- **Blend.** We use the random Gaussian noise and blend ratio 0.2 for backdoor training. The poisoning rate is 5%.
- **Adap-Blend.** We use the "hellokitty_32.png" and blend ratio of 0.2 to build triggers. The poisoning rate is 5%.
- **SSDT.** SSDT is a Source-Specific attack, which only misleads the victim classes to the targeted class. We use class 1 as the victim and class 0 as the target class.
- For other attacks and hyperparameters not mentioned, we use the default settings from the papers or their official open-source implementations.

10.4.4. DEFENSE BASELINES

BAN is compared to five representative methods: Neural Cleanse (NC) [36], Tabor [12], FeatureRE [39], Unicorn [40], and BTI-DBF [44]. NC and Tabor are designed for input

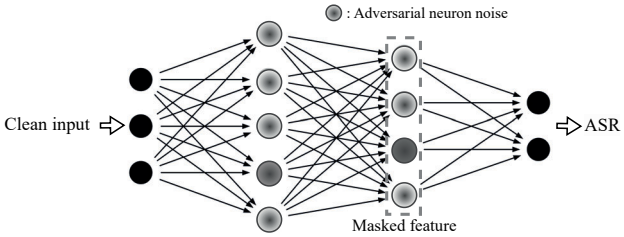


Figure 10.4: Illustrative diagram of BAN

space attacks, while the other three are designed for feature space and dealing with the latest advanced attacks. BAN uses only 1% and 5% of training data for detection and defense, respectively. Note the data used for our defense is not used for training models, i.e., the defender has no knowledge of the model to be defended. The following hyperparameters are used:

- **NC and Tabor:** We use the implementation and default hyperparameters from TrojanZoo [27] for NC and Tabor. 1% of the training set is used to conduct 100 epochs of the trigger inversion. The learning rate is 0.01.
- **FeatureRE:** We first tried the default hyperparameters from the FeatureRE paper, but they could not work even for BadNets. We assume this is because the constraints, including feature mask size and similarity between original images and trigger images, are limited to a very small size. Therefore, we relaxed the constraints on masks and norms to find stronger triggers, including loss std bound=1.2, p loss bound=0.2, loss std bound=1.2, mask size=0.06, and learning rate=0.001. We use 1% of the training set and run FeatureRE for 400 epochs for each class.
- **Unicorn:** The default hyperparameters for Unicorn are too powerful in our case, and the recovered triggers can mislead any model to the target label when applied to inputs. Every model, including benign models, is thought of as backdoored. Therefore, we slightly increase thread values of the constraints on Unicorn optimization to find proper triggers, including loss std bound=0.5, SSIM loss bound=0.1, and mask size=0.01. We use 1% of the training set and run Unicorn 40 epochs for each class.
- **BTI-DBF:** We follow the default settings in the BTI-DBF [44] paper. Note that BTI-DBF uses 5% of training samples for defenses.

In this section we use bold font to denote the best results.

10.4.5. BAN SETTINGS

Detection. We use epsilon (ϵ) to limit the noise added to neurons. Otherwise, the model weights are destroyed by the noise. The ϵ values are 0.3, 0.3, 0.2, and 0.1 for CIFAR-10, GTSRB, Tiny-ImageNet, and ImageNet200, respectively. ϵ values are decided according

to the size of the images. We use smaller ϵ for larger datasets. We use the 30-step PGD algorithm to solve the optimization in Eq. (10.4) to find the noise, i.e., $\boldsymbol{\delta}$ and $\boldsymbol{\xi}$. We use SGD and the learning rate of $\epsilon/30$ for the PGD optimization. Then, we use Adam and the learning rate of 0.01 to search for 20 epochs for the feature mask using Eq. (10.5). The λ_1 for Eq. (10.5) equals 0.75. The two optimizations above use 1% of the training set. Note that this 1% of data is not used to train the backdoor models. The elements in the mask are clamped into continuous values between 0 and 1. In Figure 10.4, we present an illustrative diagram of BAN.

Defense. Our defense uses Eq. (10.8) to remove the backdoor. Due to limited access to benign samples and time consumption, we only use 5% of training data and 25 epochs for our fine-tuning. The trade-off hyperparameter (λ_2) is 0.5. Then, we use the most commonly used hyperparameters for the optimizer. We use SGD with momentum=0.9, weight decay=5e-4, and learning rate=0.005 as the optimizer. The plain fine-tuning uses the same hyperparameters as BAN.

Table 10.12 shows the ablation results on fine-tuning epochs, λ_2, and learning rate. Considering both performances on benign accuracy and removing the backdoor, our hyperparameters (LR=0.005, λ_2=0.5 and epoch=25) show the best results.

Hardware. All experiments are run on a single machine with 4 RTX A6000 (48GB) and 4 RTX A5000 (24GB) GPUs, CUDA 12.0.

10.5. EXPERIMENTAL RESULTS

10.5.1. THE PERFORMANCE OF BACKDOOR DETECTION

Main Results. In Table 10.1, BAN shows better results on CIFAR-10 than all baselines, especially on advanced attacks. We note that the advanced detection methods (FeatureRE, Unicorn, and BTI-DBF) perform worse than NC on simple attacks (BadNets and Blend). We hypothesize this is because the backdoor features generated by BadNets are less obvious on feature channels than advanced attacks in the feature space, such as WaNet and IAD. Figure 10.3 shows that BadNets features are weaker than features from advanced attacks using the feature mask in Eq. (10.5). Specifically, we optimize the feature mask to disentangle the benign and backdoor features for four attacks. Then, we compute two cross-entropy loss values using the positive mask ($\mathbf{m}$) and the negative mask ($1-\mathbf{m}$) for benign and backdoor features, respectively. The average loss values and standard deviation for four models for each attack are plotted in Figure 10.3. The negative loss value of BadNets is much smaller than others, which means BadNets features are weaker than others with regard to misleading the model to the backdoor target. Recent defenses usually add more regularizers to their losses and optimization objectives to counteract powerful backdoor attacks. These regularizers encourage the capturing of strong features but omit weak ones. Thus, recent advanced detections can perform worse on BadNets than NC.

Time consumption. BAN is efficient and scalable as we do not iterate over all target classes. Figure 10.5 demonstrates that BAN uses substantially less time than all baselines. BAN is also more scalable for larger architectures or datasets. BIT-DBF (76.90s) is 1.37× slower than BAN (55.95s) on CIFAR-10 with ResNet18, and BIT-DBF (5,792.51s) without pre-training is 5.11× slower than ours (1,132.85s) on ImageNet200 with ResNet18. Fea-

Table 10.1: The detection results under different model architectures on CIFAR-10. The "Bd." refers to the number of models the defense identifies as backdoored. The "Acc." refers to detection success accuracy. The best results are marked in bold. BTI-DBF* refers to an improved version (details in Section 10.3.4).

Model	Attack	NC		Tabor		FeatureRE		Unicorn		BTI-DBF*		Ours	
		Bd.	Acc.	Bd.	Acc.	Bd.	Acc.	Bd.	Acc.	Bd.	Acc.	Bd.	Acc.
ResNet18	No Attack	0	**100%**	0	**100%**	2	90%	6	70%	0	**100%**	0	**100%**
	BadNets	20	**100%**	20	**100%**	14	70%	18	90%	18	90%	20	**100%**
	Blend	20	**100%**	20	**100%**	20	**100%**	19	95%	20	**100%**	18	90%
	WaNet	11	55%	8	40%	15	75%	20	**100%**	18	90%	20	**100%**
	IAD	0	0%	0	0%	15	75%	11	55%	20	**100%**	20	**100%**
	Bpp	0	0%	1	5%	12	60%	17	85 %	20	**100%**	20	**100%**
VGG16	No Attack	0	**100%**	0	**100%**	3	85%	6	70%	6	70%	0	**100%**
	BadNets	18	90%	16	80%	13	65%	16	80%	18	90%	19	**95%**
	Blend	19	**95%**	19	**95%**	16	80%	18	90%	16	80%	17	85%
	WaNet	10	50%	9	45%	12	60%	18	90%	16	80%	20	**100%**
	IAD	0	0%	0	0%	8	40%	17	85%	20	**100%**	20	**100%**
	Bpp	9	45%	10	50%	5	25%	15	75%	14	70%	18	**90%**
DenseNet121	No Attack	0	**100%**	0	**100%**	5	75%	8	60%	3	85%	0	**100%**
	BadNets	18	90%	20	**100%**	19	95%	15	75%	17	85%	20	**100%**
	Blend	20	**100%**	20	**100%**	12	60%	18	90%	19	95%	20	**100%**
	WaNet	13	65%	10	50%	20	**100%**	17	85%	14	70%	19	95%
	IAD	0	0%	0	0%	14	70%	16	80%	14	70%	19	**95%**
	Bpp	0	0%	0	0%	16	80%	8	40%	16	80%	20	**100%**
Average		60.56%		59.17%		72.5%		78.61%		86.39%		**97.22%**	

tureRE (297.74s) is 5.32× slower than ours on CIFAR-10 with ResNet18, and it (6,053.62s) is 45.14× slower on CIFAR-10 with DenseNet121 than ours (134.10s).

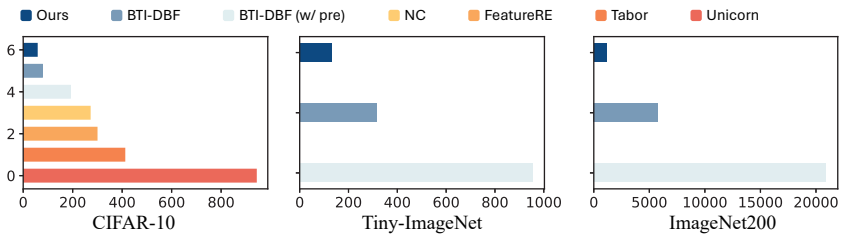


Figure 10.5: Time consumption of detection baselines on ResNet18 (in seconds) for all three datasets. BAN uses significantly less time than the baselines.

Table 10.3 shows the detection results using two larger datasets, Tiny-ImageNet and Imagenet200 and Table 10.4 shows the results on GTSRB. ImageNet200 is a subset of ImageNet-1K. We train 10 models for each case, 60 models in total. BAN still performs well under the two larger datasets. We do not apply other baselines (except for BTI-DBF) to larger datasets due to high computational requirements. For example, FeatureRE costs more than 2 hours for detection in one class of ImageNet200 (default settings), meaning FeatureRE needs more than 400 hours for a full detection on ImageNet200. In

Table 10.2: The detection results of BTI-DBF and BTI-DBF* using ResNet18 and CIFAR-10.

Attack	BTI-DBF		BTI-DBF*		Ours	
	Bd.	Acc.	Bd.	Acc.	Bd.	Acc.
No Attack	0	**100%**	0	**100%**	0	**100%**
BadNets	5	25%	18	90%	20	**100%**
Blend	0	0%	20	**100%**	18	90%
WaNet	7	35%	18	90%	20	**100%**
IAD	19	95%	20	**100%**	20	**100%**
Bpp	20	**100%**	20	**100%**	20	**100%**

GTSRB (Table 10.4), NC and BTI-DBF* perform worse against advanced attacks, while BAN precisely detects all backdoor models.

Table 10.3: The detection results using ResNet18 under different datasets.

Dataset	Attack	BTI-DBF		BTI-DBF*		Ours	
		Bd.	Acc.	Bd.	Acc.	Bd.	Acc.
Tiny-ImageNet	No Attack	0	100%	0	100%	0	100%
	WaNet	4	40%	6	60%	10	100%
	IAD	9	90%	8	80%	10	100%
ImageNet200	No Attack	3	70%	0	100%	0	100%
	WaNet	6	60%	9	90%	10	100%
	IAD	10	100%	10	100%	10	100%

Table 10.4: The detection results on GTSRB with ResNet18.

Attack	NC		BTI-DBF*		Ours	
	Bd.	Acc.	Bd.	Acc.	Bd.	Acc.
BadNets	20	100%	20	100%	20	100%
WaNet	8	40%	18	90%	20	100%
IAD	0	0%	20	100%	20	100%
Bpp	13	65%	14	70%	20	100%

10.5.2. THE PERFORMANCE OF BACKDOOR DEFENSE

A complete backdoor defense framework should include both detection and defense. The goal of defense is to decrease the attack success rate (ASR) of backdoor triggers. Detection before the defense is also necessary because the defense usually decreases the performance on benign inputs [47, 44, 39, 46, 48]. In Section 10.3.5, we propose a simple and effective fine-tuning method using the noise that activates the backdoor. Table 10.5 compares BAN with three baselines: plain fine-tuning, FeatureRE [39], and BTI-DBF(U) [44]. Plain fine-tuning refers to training the backdoor model using the same hyperparameters as BAN but without the noise loss in Eq. (10.8). The FeatureRE [39] and BTI-DBF(U) [44] refer to the defense methods from the respective paper. We use default

Table 10.5: Defense against 5 attacks using ResNet18. BA refers to benign accuracy on clean data.

Attack	No defense		Fine-tuning		FeatureRE		BTI-DBF(U)		Ours	
	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR
BadNets	93.37	99.41	92.93	87.81	**93.15**	99.79	91.26	13.12	92.06	**1.97**
Blend	94.60	100.00	93.07	99.99	**93.20**	39.28	91.86	100.00	92.72	**4.10**
WaNet	93.57	99.37	93.05	1.10	**93.67**	**0.03**	90.30	4.89	92.05	0.91
IAD	93.17	97.88	**94.11**	0.46	92.73	**0.0**	89.54	1.59	92.78	1.48
Bpp	94.29	99.93	93.85	4.46	**94.21**	98.13	90.61	2.73	92.54	**2.58**
Average	93.80	99.32	**93.40**	38.76	93.39	47.45	90.71	24.47	92.43	**2.21**

Table 10.6: Defense of backdoor attacks on Tiny-ImageNet and ImageNet200 using ResNet18.

Dataset	Attack	No defense		Fine-tuning		BTI-DBF(U)		Ours	
		BA	ASR	BA	ASR	BA	ASR	BA	ASR
Tiny-ImageNet	WaNet	58.32	99.85	**51.53**	1.3	39.49	0.96	50.69	**0.86**
	IAD	58.54	99.32	**51.86**	1.72	38.79	**0.60**	50.04	0.76
	Bpp	60.63	99.89	**57.72**	0.15	46.84	0.40	57.66	**0.10**
ImageNet200	WaNet	77.01	99.74	66.71	0.78	63.47	1.0	**69.95**	**0.58**
	IAD	76.72	99.75	69.91	**0.42**	64.33	1.24	**72.18**	1.30
	Bpp	78.56	99.88	70.89	**0.82**	67.02	3.10	**72.59**	2.68

hyperparameters for FeatureRE [39] and BTI-DBF(U) [44]. For plain fine-tuning and BAN, we use a small learning rate (0.005) to avoid jumping out of the current optimal parameters of the well-trained model. Then, we use λ_2 (0.5) for Eq. (10.8) for the trade-off between robustness against backdoor and clean accuracy. We fine-tune for a short schedule of 25 epochs, as the model is well-trained. Table 10.12 in Section 10.4.5 shows defense performance with standard deviation on different hyperparameters, which supports our choice. Tables 10.5 and 10.6 demonstrate that our fine-tuning method effectively removes the backdoor while preserving high accuracy in benign inputs. ANP [41] uses a similar neuron noise for pruning backdoor neurons. Their optimization objective includes neuron noise and neuron mask. The neuron noise is optimized to fool the network, while the mask controls a trade-off for clean cross-entropy loss. We compare our fine-tuning with ANP in Table 10.7. The hyperparameters of ANP are the default from the original paper.

10

Table 10.7: Comparison with ANP [41] on CIFAR-10 using ResNet18.

Defense	BadNets		Blend		WaNet		IAD		Bpp	
	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR
No defense	93.37	99.41	94.60	100	93.57	99.37	92.83	97.10	94.29	99.93
ANP	93.16	2.13	94.17	97.24	93.06	13.84	92.50	0.44	93.80	4.77
Ours	92.26	2.04	92.61	1.04	92.18	0.87	92.36	1.47	92.82	2.16

10.5.3. DEFENSE AGAINST ALL-TO-ALL ATTACKS

Previous works [39, 40] usually only consider all-to-one attacks, which limits the application in practical situations. In this section, we evaluate our fine-tuning method under three all-to-all attacks: WaNet-All [26], IAD-All [25], and Bpp-All [38]. FeatureRE is designed for all-to-one attacks, so we use target label 0 here for FeatureRE. FeatureRE is included to show that the all-to-one defense does not work in the all-to-all setting. BAN is capable of handling all-to-all attacks because we directly explore the neurons themselves instead of optimizing for the potential targeted label. Table 10.8 demonstrates the effectiveness of our method.

Table 10.8: Defense against all-to-all backdoor attacks. BA refers to benign accuracy on clean data.

Attack	No defense		Fine-tuning		FeatureRE		BTI-DBF(U)		Ours	
	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR
WaNet-All	93.60	91.86	92.65	18.03	**93.33**	91.96	91.30	1.72	92.29	**1.11**
IAD-All	92.96	90.62	**93.19**	3.72	93.06	91.20	91.36	3.72	92.31	**1.13**
Bpp-All	94.45	84.68	**93.90**	1.58	94.32	83.87	90.16	2.05	93.23	**1.38**
Average	93.67	89.05	93.25	7.78	**93.57**	89.01	90.94	2.49	92.61	**1.21**

10.5.4. EVALUATION UNDER ADAPTIVE ATTACK

Attackers may design a specific attack for a defense when they know its details [34]. In this section, we evaluate BAN against two attacks that attempt to bypass the difference between backdoor and benign features: Adap-Blend [28] and SSDT attack [24]. Both Adap-Blend [28] and SSDT attack [24] try to obscure the difference between benign and backdoor latent features. Adap-Blend [28] achieves it by randomly applying 50% of the trigger to poison the training data, while SSDT attack [24] utilizes source-specific and dynamic-triggers to reduce the impact of triggers on samples from non-victim classes. The source-specific attack refers to backdoor triggers that mislead the model to the target class only when applied to victim class samples. Table 10.9 demonstrates our approach is resistant to these two attacks, while other methods fail. The reason is that the backdoor features of SSDT are close to benign features in the feature space. It is difficult for other methods to distinguish between backdoor and benign features created by SSDT. Our detection method directly analyzes the model itself using neuron noise, which captures the difference between backdoor and benign models concerning parameters. Table 10.10 shows the fine-tuning results using our defense method against the two Adap-blend and SSDT. The results show that fine-tuning with neuron noise can remove the backdoor with a slight decrease in clean accuracy.

Table 10.9: The detection results under adaptive attacks on using CIFAR-10 and ResNet18.

Attack	NC		Tabor		FeatureRE		Unicorn		BTI-DBF*		Ours	
	Bd.	Acc.	Bd.	Acc.	Bd.	Acc.	Bd.	Acc.	Bd.	Acc.	Bd.	Acc.
Adap-Patch	18	90%	15	75%	17	85%	20	**100%**	20	**100%**	20	**100%**
SSDT	0	0%	0	0%	0	0%	0	0%	20	**100%**	20	**100%**

Table 10.10: Fine-tuning for adaptive attacks.

Attack	No defense		Ours	
	BA	ASR	BA	ASR
Adap-Patch	94.20	99.75	90.45	10.24
SSDT	93.86	90.30	93.29	0.90

10.5.5. ANALYSIS ON PROMINENT FEATURES

We provide additional analysis of the phenomenon that backdoor features are more prominent for advanced attacks (WaNet, IAD, and Bpp) than weaker attacks (BadNets, Blend). Table 10.11 demonstrates that previous decoupling methods cannot easily pick up backdoor features from weak attacks, such as BadNets. In particular, when detecting without the L_1 regularizer (i.e., w/o norm), the negative feature loss of BadNets is high with a very large L_1 mask norm, while the Bpp has an even higher negative loss with a much smaller mask norm. The high negative loss of BadNets is actually from the sparse feature mask rather than backdoor features, i.e., there are too many zeros in (1 - **m**). This indicates that BadNet backdoor features are less prominent than Bpp features, making it more challenging to decouple BadNets features.

10.5.6. DIFFERENT POISONING RATES

BAN is effective against BadNets with different poisoning rates, as shown in Table 10.13. We use BadNets because it is relatively weak at a low poisoning rate (i.e., hard to be mitigated), while more advanced attacks may still be strong at a low poisoning rate.

10.5.7. AGAINST BACKDOORS ON ADDITIONAL ARCHITECTURES

We evaluate the defense performance on a 4-layer MLP is trained with benign samples and with BadNets and on a 12-layer Swin transformer on CIFAR-10. Table 10.14 demonstrates that BAN is effective on MLP, where the "num. to target" refers to the number of samples (in 5000 validation samples) that are classified as the backdoor target after our detection. We also find that the positive feature loss (i.e., benign feature loss) is very close to the negative loss (i.e., potential backdoor feature loss), which indicates that the backdoor features are more challenging to decouple from benign ones.

Table 10.15 provides experimental results of a 12-layer Swin transformer on CIFAR-10. For BadNets and Blend, we train the backdoored network using Adam as the optimizer. For WaNet, IAD, and Bpp, we use the default setting. All backdoored models can be detected by BAN.

10.5.8. CHOOSING THE λ_1

As discussed in the method section, the mask norm (L_1 norm) with lambda in Eq. (10.5) is to ensure that the optimization objective is decoupling between benign and backdoor features. Without the constraint of the mask norm in Eq. (10.5), the optimization objective will be simply increasing the mask norm unless there are extremely strong backdoor features. The λ_1 value selection is implemented by checking the value of the mask norm. Table 10.16 provides an example of λ_1 selection on CIFAR-10, where the maximal mask

Table 10.11: Loss values when feeding the benign or backdoor features into the final classification layer. The mask is optimized using Eq. (10.5) (w/ norm) or Eq. (10.7) (w/o norm). The shape of the feature mask is $512 \times 4 \times 4$, so the maximal L_1 norm is 8192.

Attack	BA	ASR	L_1 mask norm	Positive feature loss	Negative feature loss
BadNets (w/ norm)	93.47	99.71	2258.90	0.21	0.26
BadNets (w/o norm)	93.47	99.71	8054.45	0.14	2.17
Blend (w/ norm)	94.60	100.00	2084.62	0.15	0.20
Blend (w/o norm)	94.60	100.00	8117.90	0.04	2.22
WaNet (w/ norm)	93.88	99.63	7400.97	0.06	2.34
WaNet (w/o norm)	93.88	99.63	7702.56	0.05	2.39
IAD (w/ norm)	93.82	99.64	7898.91	0.03	2.25
IAD (w/o norm)	93.82	99.64	7895.16	0.03	2.25
Bpp (w/ norm)	94.56	99.97	7147.68	0.09	2.80
Bpp (w/o norm)	94.56	99.97	7260.31	0.09	2.78

Table 10.12: Ablation of hyperparameters for our defense. Each entry is the average of 5 runs.

Attack	LR	Epoch	λ_2	Ours	
				BA	ASR
Bpp	0.005	25	0.2	93.26 ± 0.14	2.53 ± 0.93
	0.005	25	0.5	92.64 ± 0.19	2.28 ± 0.72
	0.005	25	0.8	92.19 ± 0.13	2.89 ± 0.94
	0.005	25	1.0	92.04 ± 0.29	2.74 ± 0.49
	0.005	10	0.5	92.97 ± 0.29	2.62 ± 0.44
	0.005	50	0.5	92.57 ± 0.09	2.80 ± 0.73
	0.005	75	0.5	92.51 ± 0.16	2.34 ± 0.17
	0.001	25	0.5	93.57 ± 0.17	2.89 ± 1.01
	0.010	25	0.5	91.99 ± 0.22	2.52 ± 0.64
	0.020	25	0.5	90.05 ± 0.37	1.91 ± 0.55

Table 10.13: The performance of BAN fine-tuning under different poisoning rates of BadNets using CIFAR-10.

Poisoning Rate	BA	ASR	pos. feature loss	neg. feature loss	Mitigated BA	Mitigated ASR
0.01	93.48	98.69	0.38	0.17	92.07	2.73
0.05	93.37	99.41	0.37	0.35	92.06	1.97
0.10	90.98	100.00	0.35	2.06	90.29	2.17
0.15	90.32	100.00	0.39	2.23	90.16	1.71
0.20	89.34	100.00	0.44	2.43	90.39	1.01
0.25	88.09	100.00	0.56	2.81	89.55	1.54
0.30	86.09	100.00	0.62	3.13	88.83	1.08
0.40	82.39	100.00	0.67	3.51	88.75	1.67
0.50	77.83	99.97	0.84	4.27	86.87	3.56

Table 10.14: The performance of BAN on the 4-layer MLP.

MLP	BA	ASR	Pos. feature loss	Neg. feature loss	Num. to target	Mitigated BA	Mtigated ASR
Benign	53.24	-	1.95	2.29	419	-	-
BadNets	47.04	100.00	2.03	2.34	3607	45.13	7.11

norm is 8192. It can be observed that the mask is almost full of ones when λ_1 is smaller than 0.7, and the negative feature loss (backdoor feature) is high, based on which we can pick a value of λ_1 greater than 0.7. Note that the selection is unaware of potential backdoors.

10.5.9. CLEAN LABEL ATTACK

We provide a clean-label backdoor experiment following [35], where we take default hyperparameters. Table 10.17 demonstrates the detection performance of 20 models, where we randomly select one model from the 20 for the mitigation experiment. Experimental results demonstrate that BAN is also effective against the clean-label attack.

10.6. DISCUSSION

Neuron Noise and Lipschitz Continuity. A small trigger that changes the output of a benign input into a malicious target label can be related to the high Lipschitz constant [45] and a neural network with high robustness tends to have a lower local Lipschitz constant. Moreover, a larger local Lipschitz constant implies steeper output around trigger-inserted points, leading to a smaller trigger effective radius making trigger inversion less effective [49]. Thus, the concepts of adversarial activation noise and Lipschitz continuity are related, and the local Lipschitz constant can serve as an upper bound for the trigger's effective influence. In addition, introducing theoretical tools like the Lipschitz constant for a backdoor defense may also be very tricky in practice because it needs approximation for implementation. For example, [45] evaluates the channel-wise Lipschitz constant by its upper bound but does not thoroughly discuss the relationship between the channel-wise Lipschitz constant and the network-wise Lipschitz constant, where the theorem of Lipschitz continuity really relies on. Recent work also mentions that empirically estimating the Lipschitz constant is hard from observed data, which usually leads to overestimation [5]. Methods relying on Lipschitz continuity may not require heavy computational

Table 10.15: Results on Swin Transformer.

Attack	No defense		FT		BTI-DBF(U)		BAN Fine-tune	
	BA	ASR	BA	ASR	BA	ASR	BA	ASR
BadNets	86.7	97.43	82.23	44.07	85.21	99.99	83.92	2.91
Blend	85.46	100.00	80.13	100.00	83.9	100.00	79.18	26.20
WaNet	78.25	31.03	75.63	2.3	73.83	2.61	75.28	3.1
IAD	76.35	88.71	74.58	54.63	74.29	61.16	76.26	9.8
Bpp	79.19	63.74	75.35	12.99	74.66	11.04	74.25	6.91

Table 10.16: Feature loss values with different λ_1

λ_1	Mask norm	Pos. feature loss	Neg. feature loss
0.0	8188.62	0.27	2.30
0.1	8188.75	0.28	2.30
0.2	8184.30	0.27	2.30
0.3	8175.50	0.29	2.30
0.4	8152.40	0.25	2.30
0.5	8131.26	0.27	2.29
0.6	8055.07	0.21	2.27
0.7	7898.25	0.26	2.24
0.8	596.85	0.99	0.23
0.9	22.08	2.33	0.28

load and are also related to our approach. Our method emphasizes more on fine-tuning with the guidance of neuron noise, rather than tuning the trained model.

Limitations. Similar to existing works [36, 21, 12, 39, 40, 44], BAN assumes a local small and benign dataset. This scenario is common, considering that benign samples are available online, and the model provider may provide some samples to verify the model's performance. In addition, our fine-tuning with neuron noise may slightly decrease the performance in benign inputs, similar to other defense methods [39, 44, 47, 41, 18, 36, 48]. However, BAN provides better performance and requires less time consumption.

Broader Impacts. This paper proposes a complete defense that includes detecting and removing DNN backdoors. We believe our approach has a positive impact on the security of DNNs. A possible negative impact is overconfidence in robustness against backdoor attacks, as there is no theoretical guarantee always to remove the backdoor.

10.7. CONCLUSIONS AND FUTURE WORK

This paper proposes an effective yet efficient backdoor defense, BAN, that utilizes the adversarial neuron noise and the mask in the feature space. BAN is motivated by the observation that traditional defenses outperformed the latest feature space defenses on input space backdoor attacks. To this end, we provide an in-depth analysis showing that feature space defenses are overly dependent on prominent backdoor features. Experimental studies demonstrate BAN's effectiveness and efficiency against various types of backdoor attacks. We also show BAN's resistance to potential adaptive attacks. Future studies could explore a more practical detection method without assuming access to local benign samples and better strategies for decoupling features because a fixed mask in the feature is not always aligned with the benign and backdoor features.

Table 10.17: Clean label attack on CIFAR-10.

Attack	BA	ASR	Detection Success Accuracy	Mitigated BA	Mitigated ASR
LC [35]	92.72	100.00	90.00	89.27	6.16

REFERENCES

[1] Gorka Abad et al. “Sneaky Spikes: Uncovering Stealthy Backdoor Attacks in Spiking Neural Networks with Neuromorphic Data.” In: *NDSS*. 2024.

[2] Eugene Bagdasaryan and Vitaly Shmatikov. “Blind Backdoors in Deep Learning Models”. In: *USENIX Security*. 2021.

[3] M. Barni, K. Kallas, and B. Tondi. “A New Backdoor Attack in CNNS by Training Set Corruption Without Label Poisoning”. In: *ICIP*. 2019.

[4] Bryant Chen et al. “Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering”. In: *SafeAI Workshop @ AAAI* (2018).

[5] Erh-Chung Chen et al. “Data-Driven Lipschitz Continuity: A Cost-Effective Approach to Improve Adversarial Robustness”. In: *arXiv preprint arXiv:2406.19622* (2024).

[6] Xinyun Chen et al. “Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning”. In: *arXiv* (2017).

[7] Jia Deng et al. “ImageNet: A large-scale hierarchical image database”. In: *CVPR*. 2009.

[8] Khoa Doan et al. “LIRA: Learnable, Imperceptible and Robust Backdoor Attacks”. In: *ICCV*. 2021.

[9] Tianyu Gu et al. “BadNets: Evaluating Backdooring Attacks on Deep Neural Networks”. In: *IEEE Access* 7 (2019).

[10] Junfeng Guo, Ang Li, and Cong Liu. “AEVA: Black-box Backdoor Detection Using Adversarial Extreme Value Analysis”. In: *ICLR*. 2022.

[11] Junfeng Guo et al. “SCALE-UP: An Efficient Black-box Input-level Backdoor Detection via Analyzing Scaled Prediction Consistency”. In: *ICLR*. 2023.

[12] Wenbo Guo et al. “Towards Inspecting and Eliminating Trojan Backdoors in Deep Neural Networks”. In: *ICDM*. 2020.

[13] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *CVPR*. 2016.

[14] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. “Handcrafted backdoors in deep neural networks”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 8068–8080.

[15] Gao Huang et al. “Densely connected convolutional networks”. In: *CVPR*. 2017.

[16] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).

[17] Ya Le and Xuan Yang. “Tiny imagenet visual recognition challenge”. In: *CS 231N* (2015).

[18] Yige Li et al. “Anti-Backdoor Learning: Training Clean Models on Poisoned Data”. In: *NeurIPS*. 2021.

[19] Yinshan Li et al. “NTD: Non-Transferability Enabled Deep Learning Backdoor Detection”. In: 2024.

[20] Yuezun Li et al. “Invisible Backdoor Attack With Sample-Specific Triggers”. In: *ICCV*. 2021.

[21] Yingqi Liu et al. “ABS: Scanning Neural Networks for Back-doors by Artificial Brain Stimulation”. In: *CCS*. 2019.

[22] Yingqi Liu et al. “Trojaning Attack on Neural Networks”. In: *NDSS*. 2018.

[23] Aleksander Madry et al. “Towards Deep Learning Models Resistant to Adversarial Attacks”. In: *ICLR*. 2018.

[24] Xiaoxing Mo et al. “Robust Backdoor Detection for Deep Learning via Topological Evolution Dynamics”. In: *SP*. 2024.

[25] Anh Nguyen and Anh Tran. “Input-Aware Dynamic Backdoor Attack”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[26] Tuan Anh Nguyen and Anh Tuan Tran. “WaNet - Imperceptible Warping-based Backdoor Attack”. In: *ICLR*. 2021.

[27] Ren Pang et al. “TrojanZoo: Towards Unified, Holistic, and Practical Evaluation of Neural Backdoors”. In: *Euro S&P*. 2022.

[28] Xiangyu Qi et al. “Revisiting the assumption of latent separability for backdoor defenses”. In: *ICLR*. 2023.

[29] Xiangyu Qi et al. “Towards a proactive ML approach for detecting backdoor poison samples”. In: *USENIX Security*. 2023.

[30] Guangyu Shen et al. “Backdoor Scanning for Deep Neural Networks through K-Arm Optimization”. In: *ICML*. 2021.

[31] Reza Shokri et al. “Bypassing backdoor detection algorithms in deep learning”. In: *Euro S&P*. 2020.

[32] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *ICLR*. 2015.

[33] Johannes Stallkamp et al. “Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition”. In: *Neural Networks* (2012), pp. 323–332.

[34] Florian Tramer et al. “On Adaptive Attacks to Adversarial Example Defenses”. In: *NeurIPS*. 2020.

[35] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. “Label-consistent backdoor attacks”. In: *arXiv preprint arXiv:1912.02771* (2019).

[36] Bolun Wang et al. “Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks”. In: *SP*. 2019.

[37] Hang Wang et al. “Mm-bd: Post-training detection of backdoor attacks with arbitrary backdoor pattern types using a maximum margin statistic”. In: *SP*. 2024.

[38] Zhenting Wang, Juan Zhai, and Shiqing Ma. "BppAttack: Stealthy and Efficient Trojan Attacks Against Deep Neural Networks via Image Quantization and Contrastive Adversarial Learning". In: *CVPR*. 2022.

[39] Zhenting Wang et al. "Rethinking the Reverse-engineering of Trojan Triggers". In: *NeurIPS*. 2022.

[40] Zhenting Wang et al. "UNICORN: A Unified Backdoor Trigger Inversion Framework". In: *ICLR*. 2023.

[41] Dongxian Wu and Yisen Wang. "Adversarial Neuron Pruning Purifies Backdoored Deep Models". In: *NeurIPS*. 2021.

[42] Jing Xu et al. "Watermarking graph neural networks based on backdoor attacks". In: *Euro S&P*. 2023.

[43] Xiaoyun Xu et al. "Universal Soldier: Using Universal Adversarial Perturbations for Detecting Backdoor Attacks". In: *DSN-W*. 2024.

[44] Xiong Xu et al. "Towards Reliable and Efficient Backdoor Trigger Inversion via Decoupling Benign Features". In: *ICLR*. 2024.

[45] Runkai Zheng et al. "Data-Free Backdoor Removal Based on Channel Lipschitzness". In: *ECCV*. 2022.

[46] Hong Zhu et al. "NeuralSanitizer: Detecting Backdoors in Neural Networks". In: *TIFS* 19 (2024), pp. 4970–4985.

[47] Mingli Zhu et al. "Enhancing Fine-Tuning Based Backdoor Defense with Sharpness-Aware Minimization". In: *ICCV*. 2023.

[48] Mingli Zhu et al. "Neural Polarizer: A Lightweight and Effective Backdoor Defense via Purifying Poisoned Features". In: *NeurIPS*. 2023.

[49] Rui Zhu et al. "Gradient Shaping: Enhancing Backdoor Attack Against Reverse Engineering". In: *NDSS*. 2024.

11

TOWARDS BACKDOOR STEALTHINESS IN MODEL PARAMETER SPACE

Backdoor attacks maliciously inject covert functionality into machine learning models, representing a security threat. The stealthiness of backdoor attacks is a critical research direction, focusing on adversaries' efforts to enhance the resistance of backdoor attacks against defense mechanisms. Recent research on backdoor stealthiness focuses mainly on indistinguishable triggers in input space *and inseparable backdoor representations in* feature space*, aiming to circumvent backdoor defenses that examine these respective spaces. However, existing backdoor attacks are typically designed to resist a specific type of backdoor defense without considering the diverse range of defense mechanisms. Based on this observation, we pose a natural question:* Are current backdoor attacks truly a real-world threat when facing diverse practical defenses?

To answer this question, we examine 12 common backdoor attacks that focus on input-space or feature-space stealthiness and 17 diverse representative defenses. Surprisingly, we reveal a critical blind spot that backdoor attacks designed to be stealthy in input and feature spaces can be mitigated by examining backdoored models in parameter space*. To investigate the underlying causes behind this common vulnerability, we study the characteristics of backdoor attacks in the parameter space. Notably, we find that input- and feature-space attacks introduce prominent backdoor-related neurons in parameter space, which are not thoroughly considered by current backdoor attacks. Taking comprehensive stealthiness into account, we propose a novel supply-chain attack called* `Grond`*.* `Grond` *limits the parameter changes by a simple yet effective module, Adversarial Backdoor Injection (ABI), which adaptively increases the parameter-space stealthiness during the backdoor injection. Extensive*

This chapter has been published as X. Xu, Z. Liu, S. Koffas, S. Picek. 2025. "Towards Backdoor Stealthiness in Model Parameter Space." *In Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25)*. Association for Computing Machinery, New York, NY, USA, 2863–2876. https://doi.org/10.1145/3719027.3744846.

experiments demonstrate that `Grond` *outperforms all 12 backdoor attacks against state-of-the-art (including adaptive) defenses on CIFAR10, GTSRB, and a subset of ImageNet. Additionally, we show that ABI consistently improves the effectiveness of common backdoor attacks. Our code is publicly available: https://github.com/xiaoyunxxy/parameter_backdoor.*

Table 11.1: A summary of the existing defenses evaluated in this paper. *"Proactively training"* refers to the strategy that the defender could proactively control the training on poisoned training data to produce a clean model without a backdoor in it. Additionally, all the defenses have been tested against the all-to-one attack, so we omitted it from the attack assumptions. A summary of backdoor attacks is provided in Table 11.6.

Defense		Defense Task			Threat Model			Attack Assumption	
		Input detection	Model detection	Mitigation	Black-box	Needs clean data	Proactively training	All-to-all	Dynamic
Model Inspection	NC [56]	⊖	●	⊖	⊖	●	⊖	⊖	⊖
	Tabor [16]	⊖	●	⊖	⊖	●	⊖	⊖	⊖
	FeatureRE [58]	⊖	●	⊖	⊖	●	⊖	⊖	●
	Unicorn [59]	⊖	●	⊖	⊖	●	⊖	⊖	●
	BTI-DBF [65]	⊖	●	⊖	⊖	●	⊖	●	●
Input Inspection	Scale-up [15]	●	⊖	⊖	●	⊖	⊖	●	●
	IBD-PSC [19]	●	⊖	⊖	⊖	●	⊖	●	●
	CT [42]	●	⊖	⊖	⊖	●	●	●	●
Pruning	FP [28]	⊖	⊖	●	⊖	●	⊖	●	●
	ANP [62]	⊖	⊖	●	⊖	●	⊖	●	●
	CLP [71]	⊖	⊖	●	⊖	⊖	⊖	●	●
	RNP [24]	⊖	⊖	●	⊖	●	⊖	●	●
Fine-tune	vanilla FT	⊖	⊖	●	⊖	●	⊖	●	●
	FT-SAM [73]	⊖	⊖	●	⊖	●	⊖	●	●
	I-BAU [67]	⊖	⊖	●	⊖	●	⊖	●	●
	FST [35]	⊖	⊖	●	⊖	●	⊖	●	●
	BTI-DBF(U) [65]	⊖	⊖	●	⊖	●	⊖	●	●

⊖ the item is not supported by the defense; ● the item is supported by the defense.

11.1. INTRODUCTION

While deep neural networks (DNNs) have achieved excellent performance on various tasks, they are vulnerable to backdoor attacks. Backdoor attacks insert a secret functionality into a model, which is activated by malicious inputs during inference. Such inputs contain an attacker-chosen property called the trigger. Backdoored DNNs can be created by training with poisoned data [14, 5, 39, 54]. More powerful and stealthy backdoors can also be injected through the control of a training process [48, 1, 7, 72, 43, 36, 70], or by direct weights modification of the victim model [18, 2].

In early backdoor attacks [14, 5, 31], triggers could induce noticeable changes that human inspectors or anomaly detectors [3, 56, 30] could easily spot. To enhance the ability to remain undetected against such defenses (i.e., achieve *input-space stealthiness*), smaller or more semantic-aware triggers are designed [39, 10, 57]. Input-space stealthy backdoor attacks usually need to change labels of poisoned samples to the target class (i.e., dirty-label), which makes detection easier [3]. To this end, another line of backdoor attacks poisons the training data without changing the labels [54, 68] (i.e., clean-label), improving backdoor stealthiness.

Despite the stealthiness concerning input images and labels, it has been widely observed that existing backdoor attacks introduce separable representations in the feature space, which can be exploited to develop backdoor defenses [58, 41, 35, 73, 65]. For example, featureRE [58] utilizes feature separability and designs a feature space constraint to reverse-engineer the backdoor trigger. In response to feature-space defenses, state-of-the-art (SOTA) backdoor attacks focus on eliminating the separability in the feature space [36, 41, 74, 48] to increase the *feature-space stealthiness*, i.e., the undetectability against feature-space defenses. Considering a different threat model, supply-chain backdoor attacks assume control over the training or directly modify the model's weights [7, 18, 2], and the backdoored model is provided as a service or as the final product. For example, supply-chain attacks could introduce a penalty to the training loss that de-

creases the distance between the backdoor and benign features to increase feature-space stealthiness [48, 70, 9, 72].

An important observation is that most backdoor attacks are designed to be stealthy to resist a specific type of defense. For example, WaNet [39] and Bpp [57] design imperceptible triggers to bypass input-space defenses (such as NC [56]), but introduce significant separability in the feature space [58]. Adap-patch [41] avoids feature separability but uses patch-based triggers, which a human inspector can detect. More critically, current backdoor attacks are barely evaluated against *parameter-space defenses* [73, 27, 64, 62, 71, 24]. This oversight is significant because backdoor behaviors are ultimately embedded in and reflected by the parameters of the backdoored model, which is the final product of any backdoor attack. As such, there is a lack of systematic evaluation of backdoor attacks against the latest *parameter-space defenses*.

To this end, in this paper, we first systematically analyze 12 attacks against 17 backdoor defenses. All evaluated defenses and their characteristics, including detection and mitigation, are summarized in Table 11.1. Surprisingly, our experiments demonstrate that parameter-space defenses can easily mitigate SOTA stealthy backdoor attacks (including supply-chain attacks), indicating that existing stealthy backdoor attacks fail to provide *parameter-space stealthiness* and, as a result, still need substantial improvement to be stealthy in the model's parameter space. More importantly, our analysis reveals that even though some backdoor attacks can resist several defenses, bypassing all defense types is far from trivial.

To explore whether it is possible to make backdoor attacks stealthy simultaneously against diverse defenses, we propose a novel attack called `Grond` that considers *comprehensive stealthiness*, meaning that a backdoor attack is stealthy in the input, the feature, and the parameter space of the model. `Grond` achieves the input space stealthiness by using adversarial perturbations as the trigger. To achieve parameter-space stealthiness, we propose a novel *Adversarial Backdoor Injection* module that adaptively injects the backdoor during the backdoor training to achieve parameter space stealthiness. We also show that the feature-space stealthiness is a by-product of input- and parameter-space stealthiness with empirical results in Figures 11.7 and 11.9. Specifically, guided by our TAC analysis, we leverage the Lipschitz continuity of neuron activations to find backdoor-related suspicious and sensitive neurons. Then, we conduct pruning on these neurons to eliminate the backdoor effect. As a result, the backdoor is associated with neurons throughout the DNN rather than just focusing on a few prominent neurons after Adversarial Backdoor Injection, as illustrated in Figure 11.1.

We make the following contributions:

- We revisit SOTA backdoor attacks regarding their stealthiness, showing that most attacks are designed to increase input-space indistinguishability or/and feature-space inseparability without considering parameter-space stealthiness. Based on this finding, we examine common backdoor attacks and reveal a critical *blind spot* regarding real-world scenarios: SOTA stealthy backdoor attacks are highly vulnerable to parameter-space defenses.

- To investigate the underlying reasons behind this common vulnerability of backdoor attacks, we take a closer look at the backdoor characteristics in the param-

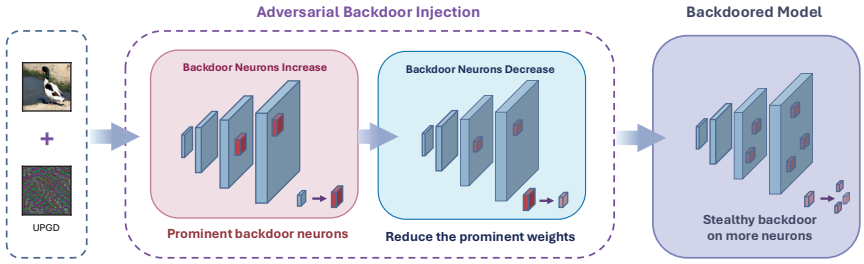


Figure 11.1: Diagram illustrating the working mechanism of `Grond`. On the left, universal PGD (UPGD) perturbation is generated as backdoor patterns to be injected. In the middle, ABI is applied where perturbed samples are iteratively used to train the model, and the model parameters are pruned to limit the magnitude of prominent backdoored weights. On the right, the output backdoored model that considers comprehensive stealthiness is deployed, where 1) the triggers are invisible, 2) the features of trigger samples are inseparable, and 3) the backdoored model weights are hardly distinguishable from benign model weights. Perturbations generated by UPGD are scaled up 10× for visualization.

eter space, showing that input- and feature-space attacks introduce prominent backdoor-related neurons, which cannot be avoided by current backdoor attacks.

- To accomplish comprehensive stealthiness, we propose a novel backdoor attack, `Grond`, that considers input, feature, and parameter-space defenses. Extensive experiments demonstrate that `Grond` outperforms SOTA attacks against four pruning- and five fine-tuning-based defenses on CIFAR10, GTSRB, and ImageNet200. Moreover, we demonstrate that `Grond` is resistant against five model detection defenses, two input detection defenses, and a proactive defense.

- We verify the effectiveness of the Adversarial Backdoor Injection module by binding it with other attacks. Experimental results demonstrate that Adversarial Backdoor Injection could substantially improve the parameter-space robustness of most common backdoor attacks.

11.2. BACKGROUND & RELATED WORK

11.2.1. PRELIMINARIES ON BACKDOOR TRAINING

This paper considers a C-class classification problem with an L-layer CNN $f = f_L \circ \cdots f_1$. Suppose that $\mathscr{D} = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{N}$ is the original training data, containing N samples of $\boldsymbol{x_i} \in \mathbb{R}^{d_c \times d_h \times d_w}$ and its label $y \in \{1, 2, \ldots, C\}$. d_c, d_h, and d_w are the number of input channels, the height, and the width of the image, respectively. The attacker chooses a target class t and creates a partially poisoned dataset $\mathscr{D}_p$ by poisoning generators G_x and G_y, i.e., $\mathscr{D}_p = \mathscr{D}_c \cup \mathscr{D}_b$. $\mathscr{D}_c$ is the benign data from original dataset, $\mathscr{D}_b = \{(\boldsymbol{x'}, y') | \boldsymbol{x'} = G_x(\boldsymbol{x}), y' = G_y(y), (\boldsymbol{x}, y) \in \mathscr{D} - \mathscr{D}_c\}$. In the clean-label setting, $G_y(y) = y$. For the dirty-label attacks, $G_y(y) = t$. In the training stage, the backdoor is inserted into f by minimizing the loss on $\mathscr{D}_p$:

$$\min_{\boldsymbol{theta}} \mathscr{L}_{\mathscr{D}_p}(\boldsymbol{theta}) = \mathop{\mathbb{E}}_{(\boldsymbol{x}, y) \in \mathscr{D}_p} \ell(f(\boldsymbol{x}; \boldsymbol{theta}), y). \tag{11.1}$$

In the inference stage, the trained f performs well on benign data $\hat{\boldsymbol{x}}$, but predicts $G_x(\hat{\boldsymbol{x}})$ as $G_y(\hat{y})$.

11.2.2. BACKDOOR ATTACKS

Backdoor attacks compromise the integrity of the victim model so that the model performs naturally on benign inputs but is misled to the target class by inputs containing the backdoor trigger. The trigger can be a visible pattern inserted into the model's input in the **input space** or a property that affects the feature representation of the model's input in the **feature space**. Eventually, however, the backdoored model's parameters in the **parameter space** will be altered regardless of the exact backdoor attack (see Figure 11.2). To insert a backdoor, the attacker is assumed to only control a small portion of the training data under the *poison training* scenario [14, 5, 69]. In the *supply-chain* setting (backdoor models provided to users), the attacker also controls the training process [48, 38, 1, 39, 57]. Moreover, the backdoor can also be created by directly modifying the model's weights [29, 18, 43, 2].

Input-space attacks. Traditional attacks typically use simple patterns as their triggers. For example, BadNets [14] uses a fixed patch, and Blend [5] mixes a Hello Kitty pattern into the images as the trigger. These non-stealthy triggers introduce abnormal data into training data and can be easily detected by human inspectors or defenses [3, 56]. To improve the stealthiness, various triggers are proposed to achieve *invisibility* in the input space. IAD [38] designed a dynamic solution in which the triggers vary among different inputs. WaNet [39] proposed the warping-based trigger, which is invisible to human inspection. Although these methods successfully build invisible triggers and bypass traditional defenses [56], they still introduce separable features and can be detected by feature-space defenses [58, 65]. These input-invisible attacks can be even more noticeable than input-visible attacks (BadNet, Blend) in the feature space [64]. We conjecture this is because they have fewer modifications on input pixels than input-visible attacks. Therefore, input-invisible attacks require more influential features to achieve a successful attack.

Feature-space attacks. Knowing the vulnerability of input-space attacks against feature-space defenses, backdoor attacks are improved for feature-space stealthiness. A common threat model of this attack type is to assume additional control over the training process. For example, [48, 9, 70, 72] directly designed loss functions to minimize the difference between the backdoor and benign features. Aside from design loss penalties, TACT [52] and SSDT [36] point out that source-specific (poison only the specified source classes) attack helps to obscure the difference in features between benign and backdoor samples. In addition, [41] proposed Adap-blend and Adap-patch, which obscures benign and backdoor features by 1) including poisoned samples with the correct label, 2) asymmetric triggers (using a stronger trigger at inference time), and 3) trigger diversification (using diverse variants of the trigger during training). Unfortunately, existing attacks lack systematic evaluation against the latest defenses. For example, Adap-blend can be thoroughly mitigated by recent works [73, 65, 64]. In summary, feature-space attacks usually introduce visible triggers and cannot defeat the latest defenses.

Supply-chain attacks. Supply-chain attacks are getting more attention due to their potential in real-world applications where backdoored models are provided as the final

product to users. In supply-chain attacks, adversaries could control both training data and the training process. Note that feature-space attacks [48, 26, 9, 70, 46, 7, 72, 63, 36] with the assumption of control over the training process are a subset of supply-chain attacks, as their output is the backdoor model. In addition to training control, another kind of supply-chain attack directly adjusts the model's weights in parameter space to introduce a backdoor, i.e., *parameter-space attack*. T-BFA [45], TBT [44], and ProFlip [4] explore modifying a sequence of susceptible bits of DNN parameters stored in the main memory (e.g., DRAM) to inject the backdoor. SRA [43] and handcrafted Backdoor [18] directly modify a subset of models' parameters to increase the logits of the target class. However, these attacks require a local benign dataset to guide the search for the subset of parameters to be modified. Data-free backdoor [33] releases the requirement of benign data by collecting substitute data irrelevant to the main task and fine-tuning using the substitute data. DFBA [2] further proposes a retraining-free and data-free backdoor attack by injecting a backdoor path (a single neuron from each layer except the output layer) into the victim model. In summary, supply-chain attacks focus on increasing the backdoor's effectiveness without comprehensively considering parameter-space defenses.

11.2.3. BACKDOOR DEFENSES

Backdoor defenses can be classified into detection and mitigation. Detection refers to determining whether a model is backdoored (*model detection*) [56, 30, 70, 59, 65] or a given input is applied with a trigger (*input detection*) [13, 15, 36]. Mitigation refers to erasing the backdoor effect from the victim model by pruning the backdoor-related neurons (*pruning-based* defenses) [28, 62, 71, 24] or unlearning the backdoor trigger (*fine-tuning-based* defenses) [73, 67, 35, 65]. In addition, recent works [23, 42, 60] also consider the home-field advantage[1] to design more powerful *proactive defenses*.

Backdoor detection. Backdoor trigger reverse engineering (also known as trigger inversion) is considered one of the most practical defenses for backdoor detection as it can be applied to both poisoning training and supply-chain scenarios [58, 59, 65, 64], i.e., it is a post-training method. Specifically, trigger inversion works by searching for a potential backdoor trigger for a specific model. The model is determined as backdoored if a trigger is found, and the trigger can be used to unlearn the backdoor. The searching is implemented as an optimization process corresponding to the model and a local benign dataset. For example, NC [56] firstly proposes trigger inversion for detection by optimizing the mask and pattern in the input space that can mislead the victim to the target class. This optimization is repeated for all classes. The model is considered backdoored if an outlier significantly smaller than the triggers for all other classes exists. Although methods similar to NC perform well against fixed patch trigger attacks, such as BadNets [14] and Blend [5], they may not be effective against input-stealthy attacks like WaNet [39]. To address this problem, FeatureRE [58] moves trigger inversion from input space to feature space. Unicorn [59] further proposes a transformation function for attacks in other spaces, such as numerical space [57]. Recent works [65, 64] focus on exploring new optimization objectives that address the inefficiency problem of previous trigger inversion methods due to optimization over all classes. BTI-DBF [65] trains a trigger generator by maximizing the backdoor feature difference between benign samples and their generated version (by

[1] The defender has full control of the system and could access the training process.

the trigger generator) and minimizing the benign feature difference. BAN [64] optimizes the noise on neuron weights rather than input pixels to activate the potential backdoor, which further improves both effectiveness and efficiency.

Backdoor mitigation. Backdoor mitigation consists of fine-tuning and pruning, which are effective and do not assume knowledge of backdoor triggers. Pruning methods aim to find and remove backdoor-related neurons. FP [28] eliminates dormant neurons on benign inputs and then fine-tunes the pruned network. ANP [62] searches for backdoor-related neurons by adding adversarial noise to neuron weights to activate the backdoor. RNP [24] uses an unlearning and recovering process on benign data to expose backdoor neurons, as the recovering will force the backdoor neurons to be silent for the main benign task. Unlike these pruning methods guided by benign data, CLP [71] directly analyzes the Channel Lipschitzness Constant of the network and prunes the high Lipschitz constant channels in a data-free manner.

Traditional fine-tuning as a defense usually needs trigger inversion methods to recover the trigger and then unlearn the trigger. For example, BTI-DBF(U) [65] fine-tunes backdoor models using triggers recovered by their inversion algorithm. However, there is no guarantee that the recovered trigger is the true trigger for the backdoor. Recent works also consider fine-tuning without the trigger information but with prior human knowledge. For example, FT-SAM [73] observes a positive correlation between the weight norm of neurons and backdoor-related neurons. Then, they propose a fine-tuning method to revise the large outliers of weight norms using Sharpness-Aware Minimization (SAM). I-BAU [67] forms a min-max fine-tuning similar to adversarial training, where the inner maximizing searches for perturbations that mislead the model, and the outer minimizing is to keep the model's capability on benign data. FST [35] assumes the backdoor and benign features should be disentangled and actively shifting features while fine-tuning by encouraging the discrepancy between the original backdoor model and the fine-tuned model.

Proactive defense. Several methods have been proposed to exploit the home-field advantage, i.e., a stronger defender, for better defensive performance. ABL [23] proposes two techniques to avoid learning the backdoor task while training on the poisoned data: 1) trapping the loss value of each example around a certain threshold because backdoor tasks are learned much faster than the main task, and their loss decreases much faster. The samples with lower loss are recorded as poisoned samples; 2) unlearning the backdoor with the recorded poisoned samples. CT [42] detects poisoned samples in the training set by introducing confusing batches of benign data with randomly modified labels. The confusing batches with random labeling corrupt the benign correlations between normal semantic features and semantic labels, so the inference model trained with confusing batches and the poisoned dataset will find it hard to distinguish benign samples. However, the correlation between the backdoor trigger and the target label remains intact, as the confusing batches contain no trigger. Therefore, samples with correctly predicted labels by the inference model are considered poisoned. PDB [60] proactively injects a defensive backdoor into the model during training, overriding the potential backdoor injected by the poisoned training data. In summary, proactive defenses assume a stronger defender for better defensive performance.

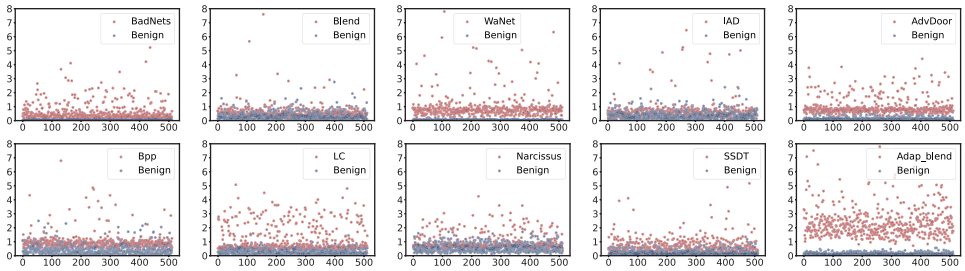


Figure 11.2: TAC [71] analysis of different backdoor attacks on 512 neurons. The y axis contains the TAC values, and the x axis depicts the index of neurons. Higher TAC values suggest a stronger relation between corresponding neurons and the backdoor trigger. In Section 11.5.6, we also provide sorted TAC value plots, clearly showing the prominent TAC values.

11.3. COMPREHENSIVE BACKDOOR STEALTHINESS

11.3.1. THREAT MODEL

Attacker's goal. The attacker provides pre-trained models to users. The aim is to inject backdoors into the pre-trained model so that the model performs well on clean inputs but predicts the attacker-chosen target label when receiving inputs with a backdoor trigger, i.e., an all-to-one attack.

Attacker's knowledge. The attacker has white-box access to the training processes, the training data, and the model weights, i.e., the supply-chain threat model. During inference, the backdoor trigger is imperceptible to human inspectors.

Attacker's capabilities. The attacker can train a well-performed surrogate model to generate UPGD, which is used to perturb the victim model's input. Additionally, the attacker can alter the model's weights during training. Table 11.6 shows that the threat model of `Grond` is aligned with baseline attacks.

11.3.2. LACK OF PARAMETER-SPACE STEALTHINESS

As introduced in the related work, early backdoor attacks that introduce noticeable changes in either input [14, 5] or feature space [39, 38] have been empirically shown powerful, even with very low poisoning rates [14, 68]. Focusing on the backdoor-introduced noticeable changes, backdoor defenses are improved to distinguish backdoor patterns in either input or feature space [58, 27]. Meanwhile, backdoor attacks are optimized to increase stealthiness in input [39] or feature space [41]. However, regardless of the implementation of input- or feature-space attack logic, backdoor behaviors are eventually embedded in the backdoored model's parameters. For this reason, it is important to investigate whether backdoor attacks introduce visible changes in the parameter space of the attacked models that can be used by the parameter-space defenses.

Considering this observation, we ran an initial experiment to understand the behavior of neurons in backdoored models. We use the TAC values [71] to quantify the relevance of a neuron to the backdoor behavior. The TAC values show the change in the output of each neuron (a feature channel of convolutional layers) before and after the trigger is attached to the input. Thus, TAC quantifies a neuron's sensitivity to the backdoor trigger. A high

TAC value indicates that the neuron is strongly related to the backdoor behavior, whereas a low TAC value shows it is not. TAC takes the exact trigger information into account when measuring the backdoor effect, which makes TAC a straightforward and effective method that captures the relevant backdoor neurons' behaviour. Specifically, TAC is defined as:

$$\mathrm{TAC}_l^{(k)}(\mathscr{D}_c) = \frac{1}{|\mathscr{D}_c|} \sum_{\boldsymbol{x} \in \mathscr{D}_c} ||f_l^{(k)}(\boldsymbol{x}) - f_l^{(k)}(G_x(\boldsymbol{x}))||_2, \tag{11.2}$$

where $f_l^{(k)}$ is the k_{th} channel of the l_{th} layer. $G_x(\boldsymbol{x})$ is the poisoned sample. $\mathscr{D}_c$ consists of a few benign samples. Note that TAC can only be used to analyze backdoor behaviors and cannot be deployed as a practical defense, as it requires access to backdoor triggers, which is unrealistic in practice.

TAC analysis of different backdoor attacks is shown in Figure 11.2, where each dot represents the TAC value for one of the 512 individual neurons. We can observe that the TAC values of neurons of backdoored models are substantially higher than those of benign models. In particular, neurons with higher TAC values contribute more to the backdoor behavior. The working mechanism of pruning- and fine-tuning-based backdoor defenses can be understood as targeting and eliminating neurons with TAC values that are substantially higher than those of others. Our observations from the TAC analysis suggest that backdoor attacks are designed to be stealthy in input space, and feature space can, in fact, be identified in parameter space, making them susceptible to parameter-space defenses. Our experimental analysis further substantiates this assumption (see Section 11.4). Thus, we conclude that current backdoor attacks may not be robust against parameter-space defenses.

11.3.3. `Grond` FOR COMPREHENSIVE STEALTHINESS

To address the vulnerabilities identified in the parameter space, we propose a stealthy backdoor attack, `Grond`, that considers comprehensive stealthiness, i.e., stealthiness in input, feature, and parameter space. `Grond` includes two key parts: UPGD trigger generation and Adversarial Backdoor Injection (ABI).

Backdoor trigger generation for input-space stealthiness. We use imperceptible adversarial perturbations to generate *imperceptible* backdoor triggers inspired by adversarial example studies [37, 69]. We modify the original PGD algorithm to generate a universal PGD (UPGD) perturbation as the backdoor trigger. UPGD contains non-robust but generalizable semantic information [53], which correlates with the benign functions of the victim model and shortens the distance between poisoned data and the target classification region [69]. Consequently, backdoor patterns tend to make fewer prominent changes to the victim network.

Similar to [68, 69], UPGD is generated on a well-trained surrogate model trained on the clean training set. The architecture and parameters of the surrogate model do not necessarily need to be the same as the victim model (see Table 11.16). UPGD is optimized following the PGD [34] algorithm to decrease the surrogate model's cross-entropy loss that takes as inputs the adversarial examples (the poisoned samples in our case) and the target class label. This procedure is described formally in Algorithm 4. The $\boldsymbol{\theta delta}$ is the generated UPGD that will be used as a backdoor trigger; thus, $G_x(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{delta}$. S is the ball function with the radius ϵ, and the small ϵ guarantees the imperceptibility of the

Algorithm 4 UPGD Generation Algorithm

Input: Surrogate model $f_{\theta_{\text{sur}}}$, training data $\mathscr{D}_c$, perturbation budget ϵ, number of iterations $\mathscr{I}$, target class t

Output: UPGD $\boldsymbol{\delta}$

$S \leftarrow B(\boldsymbol{\delta};\epsilon) = \left\{\boldsymbol{\delta} \in \mathbb{R}^{d_c \times d_h \times d_w} : \|\boldsymbol{\delta}\|_\infty \leq \epsilon\right\}$
$\boldsymbol{\delta} \leftarrow$ `random_initialization` $\wedge\ \boldsymbol{\delta} \in S$
for $i = 0$ to $\mathscr{I} - 1$ **do**
  $\mathbf{x} \leftarrow$ `sample_batch`$(\mathscr{D}_c)$
  $\mathscr{L}_{\mathscr{D}_c}(\theta) \leftarrow \mathbb{E}_{(\mathbf{x},y)\in\mathscr{D}_c}\left[\ell\left(f_{\theta_{\text{sur}}}(\mathbf{x}+\boldsymbol{\delta};\theta), t\right)\right]$
  $\boldsymbol{\delta} \leftarrow \arg\min_{\boldsymbol{\delta}\in S} \mathscr{L}_{\mathscr{D}_c}(\theta)$
end for

backdoor trigger as it controls the perturbation's magnitude.

The backdoor is injected during training by poisoning some training data from the target class, i.e., applying the UPGD trigger to the training data. In the inference stage, our backdoor is activated by the same trigger. The motivation for our small-size trigger ($\epsilon = 8$) is imperceptibility.

Adversarial Backdoor Injection for parameter-space stealthiness. Backdoor neurons (i.e., trigger-related neurons) regularly show higher activation values for inputs that contain the trigger, which results in powerful performance [30, 58, 27]. To this end, backdoor training needs to substantially increase the magnitude of parameters of backdoor neurons [62, 24, 71], which harms the parameter-space stealthiness of backdoor attacks.

One way to find the sensitive neurons with higher activation values is to analyze the Lipschitz continuity of the network. Leveraging this fact, we introduce a novel backdoor training mechanism, *Adversarial Backdoor Injection*, to increase the parameter-space backdoor stealthiness. Specifically, each neuron's Upper bound of Channel Lipschitz Condition (UCLC [71]) is calculated, based on which the weights of these suspicious neurons are set to the mean of all neurons' weights in the corresponding layer after every training epoch. In our implementation, we use the weights before every batch normalization as the neuron weights corresponding to the channel setting in UCLC. We prune neurons by substituting their weights with the mean ones because pruning to zeros makes the training unable to converge in our experiments. Formally, the k_{th} parameter of the l_{th} layer, $\boldsymbol{\theta}_l^{(k)}$, is updated as follows:

$$\boldsymbol{\theta}_l^{(k)} := \begin{cases} \text{mean}(\boldsymbol{\theta}_l), & \sigma(\boldsymbol{\theta}_l^{(k)}) > \text{mean}(\sigma(\boldsymbol{\theta}_l)) + u \times \text{std}(\sigma(\boldsymbol{\theta}_l)) \\ \boldsymbol{\theta}_l^{(k)}, & \text{otherwise,} \end{cases} \tag{11.3}$$

where u is a fixed threshold and σ is the UCLC value of the given weights. The measure for quantifying backdoor relevance can be changed from UCLC to others, such as the distance of neuron outputs when receiving benign and backdoor inputs, where a larger distance means the neuron is more relevant to backdoor behaviors and can be pruned. We use the modified UCLC for training efficiency, as UCLC is data-free, which does not require calculation based on the outputs of neurons.

In adversarial training [34], adversarial examples are introduced during training to

Table 11.2: Pruning-based mitigations against backdoored ResNet18 on CIFAR10. BA refers to benign accuracy on clean data, ASR to attack success rate, and PR to the poisoning rate of the training set. The average drop of BA and ASR is also shown with downward arrows compared to the performance without any defense. Red marks indicate the attack failed to resist the defense with an ASR lower than 60%, and green means that the ASR is higher than 60%.

Attack	No Defense		FP [28]		ANP [62]		CLP [71]		RNP [24]		Average	
	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR
BadNets [14]	93.13	100	92.42	71.71	91.60	1.06	88.99	49.02	84.04	13.82	89.26 ↓3.87	33.90 ↓66.10
Blend [5]	94.42	100	93.08	99.99	93.57	0.33	90.3	0.54	94.63	57.98	92.89 ↓1.53	39.71 ↓60.29
WaNet [39]	93.60	99.37	92.96	4.60	91.08	0.49	91.53	2.12	92.86	3.17	92.11 ↓1.49	2.59 ↓96.78
IAD [38]	92.88	97.10	91.96	1.22	92.84	0.71	92.24	0.74	92.72	0.42	92.44 ↓0.44	0.77 ↓96.33
AdvDoor [69]	93.97	100	93.37	98.69	91.46	28.83	89.22	6.13	90.17	44.60	91.05 ↓2.92	44.56 ↓55.44
Bpp [57]	94.19	99.93	93.38	18.89	92.96	2.97	93.37	1.89	92.2	5.79	92.98 ↓1.21	7.39 ↓92.54
LC [54]	94.31	100	92.22	93.57	91.02	24.43	90.96	0.38	82.70	33.60	89.23 ↓5.08	37.99 ↓62.01
Narcissus [68]	93.58	99.64	93.49	96.54	89.76	49.18	93.19	97.82	91.10	94.59	91.88 ↓1.70	84.53 ↓15.11
Adap-blend [41]	92.74	99.67	92.06	95.50	86.48	67.73	92.49	99.62	78.63	1.56	87.42 ↓5.32	66.10 ↓33.57
SSDT [36]	93.70	90.30	93.41	0.80	93.88	0.60	93.66	1.20	93.99	3.30	93.74 ↑0.04	1.47 ↓88.83
DFST [7]	95.23	100	94.79	93.84	94.64	3.72	92.43	3.53	93.29	12.68	93.79 ↓1.44	28.44 ↓71.56
DFBA [2]	88.99	100	86.85	0.03	88.96	9.55	88.96	9.57	88.96	0.90	88.43 ↓0.56	5.01 ↓94.99
Grond (PR=5%)	93.43	98.04	93.09	99.73	91.43	94.01	93.29	87.89	91.83	85.22	92.41 ↓1.02	91.71 ↓6.33
Grond (PR=1%)	94.26	93.51	93.31	96.32	92.94	91.48	94.33	87.56	92.13	94.87	93.18 ↓1.08	92.56 ↓0.95
Grond (PR=0.5%)	94.36	92.91	93.32	90.96	93.87	84.04	94.52	86.82	91.99	84.63	93.43 ↓0.93	86.61 ↓6.30

increase the model's robustness during inference. Similarly, during the Adversarial Backdoor Injection, we use backdoor defenses to increase the resistance of backdoor attacks to parameter-space defenses. At the end of each training epoch, Adversarial Backdoor Injection prunes the trained model to decrease the weights of backdoor neurons. Iteratively, backdoored neurons spread across the whole model instead of forming a few prominent backdoor neurons, as illustrated in Figure 11.1.

Feature-space stealthiness. We hypothesize that feature-space stealthiness is a by-product of parameter-space and input-space stealthiness since the variation of feature maps is strongly correlated with model parameters and inputs. Figures 11.7 and 11.9 show that Grond can substantially increase the feature-space stealthiness.

Table 11.3: Fine-tuning-based mitigations against backdoored ResNet18 on CIFAR10.

Attack	vanilla FT		FT-SAM [73]		I-BAU [67]		FST [35]		BTI-DBF(U) [65]		Average	
	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR
BadNets [14]	91.07	43.96	92.01	2.84	92.60	76.02	92.40	13.10	91.26	13.12	91.87 ↓1.26	29.81 ↓70.19
Blend [5]	91.64	99.61	92.52	1.73	91.84	8.84	93.40	100	91.86	100	92.25 ↓2.17	62.04 ↓37.96
WaNet [39]	91.11	0.99	90.89	1.03	87.98	0.81	92.17	0.04	90.30	4.89	90.49 ↓3.11	1.55 ↓97.82
IAD [38]	90.83	2.16	92.18	2.87	88.4	15.68	91.29	0.00	89.54	1.59	90.45 ↓2.43	4.46 ↓92.64
AdvDoor [69]	91.25	68.68	92.18	1.23	89.29	16.99	91.06	99.99	90.25	100	90.81 ↓3.16	57.38 ↓42.62
Bpp [57]	91.36	3.40	91.38	1.00	92.06	6.46	93.23	26.83	90.61	2.73	91.73 ↓2.46	8.08 ↓91.85
LC [54]	90.26	88.52	91.46	1.91	85.87	5.11	91.80	13.11	90.71	4.37	90.02 ↓4.29	22.60 ↓77.40
Narcissus [68]	91.70	92.91	91.76	23.98	91.48	51.74	90.06	54.22	90.94	98.11	91.19 ↓2.39	64.19 ↓35.45
Adap-blend [41]	92.42	98.73	91.23	22.4	85.38	37.31	90.91	1.19	89.17	7.09	89.82 ↓2.92	33.34 ↓66.33
SSDT [36]	93.74	0.70	93.15	0.60	90.27	3.10	92.85	0.20	90.79	1.40	92.16 ↓1.54	1.20 ↓89.10
DFST [7]	95.01	2.07	94.70	0.00	89.75	19.11	93.06	2.66	90.41	22.34	92.59 ↓2.64	9.24 ↓90.76
DFBA [2]	86.68	10.10	86.03	5.24	85.48	100	82.76	57.62	84.49	100	85.09 ↓3.90	54.59 ↓45.41
Grond (PR=5%)	91.75	94.28	92.02	80.07	90.39	93.92	93.27	99.92	91.88	99.00	91.86 ↓1.57	93.44 ↓4.60
Grond (PR=1%)	91.41	85.52	92.83	79.17	87.89	91.34	93.21	96.59	90.66	88.69	91.20 ↓3.06	88.26 ↓5.25
Grond (PR=0.5%)	91.42	82.96	92.34	76.92	89.83	79.68	93.44	92.71	90.39	91.83	91.48 ↓2.88	84.82 ↓8.09

Table 11.4: Backdoor performance of `Grond` and baseline attacks on ImageNet200 and GTSRB.

Datasets	Attack	No Defense		FT-SAM [73]		I-BAU [67]		CLP [71]		Average	
		BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR
ImageNet200	BadNets [14]	80.65	91.03	79.89	2.21	70.28	26.06	70.74	64.86	73.64 ↓7.01	31.04 ↓59.99
	Blend [5]	80.70	95.63	80.19	0.39	76.13	30.81	80.02	23.38	78.78 ↓1.92	18.19 ↓77.44
	WaNet [39]	81.24	99.97	80.41	0.66	75.67	47.27	77.18	99.78	77.75 ↓3.49	49.24 ↓50.73
	IAD [38]	79.74	99.98	75.49	0.68	77.44	15.18	76.97	84.49	76.63 ↓3.11	33.45 ↓66.53
	AdvDoor [69]	80.72	100	79.52	98.90	74.03	61.31	77.90	100	77.15 ↓3.57	86.74 ↓13.26
	Bpp [57]	81.36	92.74	79.37	1.05	76.53	3.21	80.10	2.34	78.67 ↓2.69	2.19 ↓90.55
	Narcissus [68]	81.73	81.28	80.00	83.37	77.03	56.19	80.99	86.37	79.34 ↓2.39	75.31 ↓5.97
	SSDT [36]	75.45	100	78.19	76.00	76.26	22.00	76.02	94.00	76.82 ↑1.37	64.00 ↓36.00
	`Grond`	80.92	94.11	79.05	95.05	76.89	87.75	80.29	93.83	78.74 ↓2.18	92.21 ↓1.9
GTSRB	BadNets [14]	97.19	100	95.57	0.48	92.02	29.22	96.38	0.47	94.66 ↓2.53	10.06 ↓89.94
	Blend [5]	95.92	100	93.36	0.21	92.64	38.27	93.21	0.00	93.07 ↓2.85	12.83 ↓87.17
	WaNet [39]	98.69	99.77	92.18	0.45	91.25	0.00	90.14	18.14	91.19 ↓7.50	6.19 ↓93.58
	IAD [38]	99.08	99.65	92.72	0.10	90.11	0.35	98.08	14.63	93.64 ↓5.44	5.03 ↓94.62
	AdvDoor [69]	95.80	99.99	93.94	32.26	92.67	38.20	90.09	66.39	92.23 ↓3.57	45.62 ↓54.37
	Bpp [57]	98.69	99.93	91.27	0.00	92.61	0.23	97.16	2.29	93.68 ↓5.01	0.84 ↓99.09
	Narcissus [68]	95.60	97.18	93.61	54.55	92.87	80.74	93.99	97.60	93.49 ↓2.11	77.63 ↓19.55
	SSDT [36]	96.02	77.78	93.11	0.00	90.82	0.00	94.65	19.31	92.86 ↓3.16	6.44 ↓71.34
	`Grond`	95.83	95.36	93.80	71.84	93.13	94.30	91.28	93.19	92.74 ↓3.09	86.44 ↓8.92

11.4. EXPERIMENTAL EVALUATION

11.4.1. EXPERIMENTAL SETUP

Datasets and Architectures. We follow the common settings in existing backdoor attacks and defenses and conduct experiments on CIFAR10 [22], GTSRB [50], and a subset of ImageNet [8] with 200 classes and 1,300 images per class (ImageNet200). The primary evaluation is performed using ResNet18 [17]. Moreover, we evaluate `Grond` using four additional architectures, VGG16 [49], DenseNet121 [20], EfficientNet-B0 [51], and one recent architecture InceptionNeXt [66] (see Table 11.16). We also evaluate `Grond` with large and transformer-based models (see Table 11.17).

Attack Baselines. `Grond` is compared with 12 representative attacks: BadNets [14], Blend [5], WaNet [39], IAD [38], AdvDoor [69], BppAttack [57], LC [54], Narcissus [68], Adap-Blend [41], SSDT [36], DFST [7], and DFBA [2]. The default poisoning rate is set at 5% (of the training set) for all attacks following previous work [65, 64]. Additionally, `Grond` is evaluated under various poisoning rates to provide a thorough analysis of its effectiveness. Following related works, the training schedule for attacks is 200 epochs when using CIFAR10 and GTSRB, and 100 epochs for ImageNet200. We use 1,000 images as the validation set to select the best-performing checkpoint. Like Narcissus, our attack uses the class bird (CIFAR10) as the target class. For ImageNet200, we use the "stingray" as the target class. The `Grond` poisoning rate (ImageNet200) used for results in Table 11.4 is 0.25%. For GTSRB, we use the speed limit (50) as the target class. The `Grond` poisoning rate (GTSRB) used for results in Table 11.4 is 1.74%. AdvDoor uses the same trigger and target class as ours. More details are provided in Table 11.5. For other attacks and hyperparameters not mentioned, we use the default setting from the original papers or open-source implementations.

Defense Baselines. We evaluate our attack and baseline attacks against 17 defenses, including **4 pruning-based methods** (FP [28], ANP [62], CLP [71], and RNP [24]), **5 fine-tuning-based methods** (vanilla FT, FT-SAM [73], I-BAU [67], FST [35], and BTI-

Table 11.5: The backdoor training settings.

Config	Value
Optimizer	SGD, AdamW (InceptionNeXt)
Weight decay	5×10^{-4}
learning rate	0.01
epoch	200 (GTSRB, CIFAR10), 100 (ImageNet200)
learning rate schedule	MultiStepLR (100, 150) for CIFAR10 and GTSRB, CosineAnnealingLR for ImageNet200
poison rate	0.05
u in Equation (11.3)	3.0
BadNets trigger	3×3
Blend trigger	random Gaussian noise and blend ratio 0.2
Adap-blend trigger	"hellokitty_32.png" and blend ratio of 0.2
Narcissus trigger size	$\epsilon = 16$ for both inference and training

Table 11.6: A summary of attacks evaluated in this work.

Attack	Threat Model			Trigger Type			Trigger Strategy		Stealthy Level		
	Poisoning data	Clean Label	Access Training/Model	Patch	Blend	Dynamic	All-to-all	Source Class Specific	Input	Feature	Parameter
BadNets [14]	●	⊖	⊖	●	⊖	⊖	●	⊖	⊖	⊖	⊖
Blend [5]	●	⊖	⊖	⊖	●	⊖	●	⊖	⊖	⊖	⊖
WaNet [39]	●	⊖	⊖	⊖	⊖	●	●	⊖	●	⊖	⊖
IAD [38]	●	⊖	⊖	⊖	⊖	●	●	⊖	⊖	⊖	⊖
AdvDoor [69]	●	⊖	⊖	⊖	●	⊖	●	⊖	●	●	⊖
Bpp [57]	●	⊖	⊖	⊖	⊖	●	●	⊖	●	⊖	⊖
LC [54]	●	●	⊖	●	⊖	⊖	⊖	⊖	⊖	⊖	⊖
Narcissus [68]	●	●	⊖	⊖	●	⊖	⊖	⊖	⊖	●	⊖
SSDT [36]	●	⊖	●	⊖	⊖	●	⊖	●	⊖	●	⊖
Adap-blend [41]	●	⊖	⊖	⊖	●	⊖	⊖	⊖	⊖	●	⊖
DFST [7]	●	⊖	●	⊖	●	⊖	⊖	⊖	⊖	●	⊖
DFBA [2]	⊖	⊖	●	●	⊖	⊖	⊖	⊖	⊖	●	⊖
`Grond`	●	●	●	⊖	●	⊖	⊖	⊖	●	●	●

⊖ the item is not supported by the defense; ● the item is supported by the defense.

DBF(U) [65]), **5 backdoor model detections** (NC [56], Tabor [16], FeatureRE [58], Unicorn [59], and BTI-DBF [65]), **2 backdoor input detections** (Scale-up [15] and IBD-PSC [19]), and **a proactive detection** CT [42]. Following their default settings, BTI-DBF [65] and FP [28] use 5% of training data, and other defenses use 1% of training data for detection or mitigation. CLP is a data-free backdoor pruning tool that uses no clean data. CT has access to the complete training set without knowing which samples are poisoned and can also interact with the model during training.

ANP,[2] CLP,[3] RNP,[4] FST,[5] BTI-DBF,[6] BTI-DBF(U),[6] FeatureRE,[7] Unicorn:[8] We use the implementation and default hyperparameters from their open-source code.

[2]https://github.com/csdongxian/ANP_backdoor/tree/main
[3]https://github.com/rkteddy/channel-Lipschitzness-based-pruning
[4]https://github.com/bboylyg/RNP
[5]https://github.com/AISafety-HKUST/Backdoor_Safety_Tuning
[6]https://github.com/xuxiong0214/BTIDBF/tree/master
[7]https://github.com/RU-System-Software-and-Security/FeatureRE/tree/main
[8]https://github.com/RU-System-Software-and-Security/UNICORN

FP,[9] vanilla FT,[10] FT-SAM,[11] I-BAU:[12] We use the implementation and default hyperparameters from BackdoorBench [61]. For FT-SAM on ImageNet200, the default setting will decrease benign accuracy to 0.465, so we reduce its training schedule to 25 epochs. Please note that the experiments on CIFAR10 with FT-SAM usually converge within 20 epochs in our experiments. Thus, decreasing the training schedule is not harmful to the defense performance.

NC[13] and Tabor:[14] We use the implementation from TrojanZoo [40]. 1% training set and 100 epochs are used for trigger inversion.

Scale-up,[15] IBD-PSC:[16] We use the implementation and default hyperparameters from BackdoorBox [25].

CT:[17] We use the open-source code implementation. We reduced the number of distillation iterations to 200 for efficiency reasons.

Surrogate Models. Table 11.7 provides the hyperparameters for training surrogate models to generate UPGD.

Table 11.7: The settings for training surrogate models.

Config	Value
Optimizer	SGD, AdamW (InceptionNeXt)
Weight decay	5×10^{-4}
learning rate	0.01 (CIFAR10, GTSRB), 0.001 (ImageNet200)
epoch	200 (GTSRB, CIFAR10), 100 (ImageNet200)
learning rate schedule	MultiStepLR (100, 150) for CIFAR10 and GTSRB, CosineAnnealingLR for ImageNet200

11.4.2. MAIN RESULTS ON BACKDOOR MITIGATION

All evaluated backdoor attacks are ineffective against at least one parameter-space backdoor defense on the CIFAR10, as demonstrated in Tables 11.2 and 11.3. It suggests that common backdoor attacks designed to be stealthy in input and feature spaces are vulnerable to parameter-space defenses. Given that all backdoor behaviors are embedded in parameters of backdoored models, this finding suggests that future backdoor attacks should consider parameter-space defenses as a standard step to evaluate comprehensive stealthiness.

Not surprisingly, `Grond` performs better than all baseline attacks when considering evaluated backdoor defenses since `Grond` is designed to consider comprehensive stealthiness. On four pruning-based mitigations, `Grond` achieves 7.18% higher ASR on

[9] https://github.com/SCLBD/BackdoorBench/blob/main/defense/fp.py
[10] https://github.com/SCLBD/BackdoorBench/blob/main/defense/ft.py
[11] https://github.com/SCLBD/BackdoorBench/blob/main/defense/ft-sam.py
[12] https://github.com/SCLBD/BackdoorBench/blob/main/defense/i-bau.py
[13] https://github.com/ain-soph/trojanzoo/blob/main/trojanvision/defenses/backdoor/model_inspection/neural_cleanse.py
[14] https://github.com/ain-soph/trojanzoo/blob/main/trojanvision/defenses/backdoor/model_inspection/tabor.py
[15] https://github.com/THUYimingLi/BackdoorBox/blob/main/core/defenses/SCALE_UP.py
[16] https://github.com/THUYimingLi/BackdoorBox/blob/main/core/defenses/IBD_PSC.py
[17] https://github.com/Unispac/Fight-Poison-With-Poison

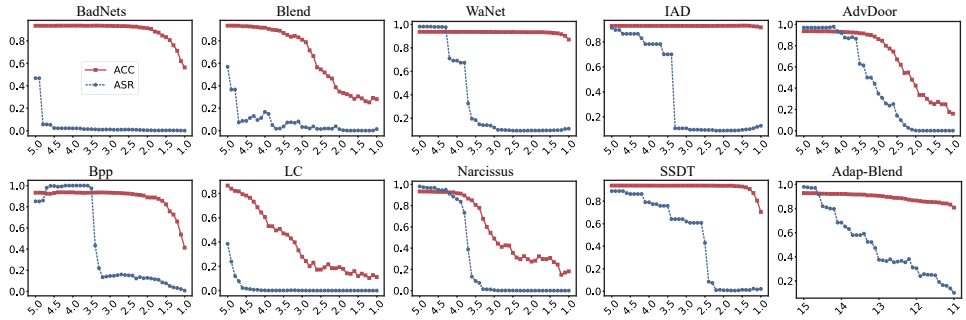


Figure 11.3: Pruning neurons with high TAC values using different thresholds (the x axis).

average than the best backdoor attack, Narcissus. On five fine-tuning mitigations that show more powerful defense capability than pruning-based mitigations, `Grond` achieves 29.25% higher ASR on average than Narcissus. In addition, `Grond` bypasses the five model detection and two input-space detections (see Section 11.4.6).

Pruning-based mitigation. We take a closer look at the details of pruning-based backdoor mitigation experiments in Table 11.2, presenting the results of all attacks against four pruning-based defenses. BadNets and Blend perform better on average than input-space stealthy attacks, e.g., WaNet and Bpp, because input-space stealthy attacks introduce significant separability in the feature space (see Figures 11.7 and 11.9). Across all pruning-based defenses, FP performs the worst, as expected, since it follows regular model pruning practice and is not a tailored backdoor pruning method.

Fine-tuning-based mitigations. Table 11.3 presents the backdoor performance against five fine-tuning-based defenses. In general, fine-tuning-based defenses are more effective than pruning-based defenses. For example, Narcissus and Adap-Blend can achieve ASRs higher than 60% against three out of four pruning-based defenses but are much less effective against most fine-tuning-based methods. FT-SAM is the most effective across all defenses, as shown in Tables 11.2 and 11.3, being able to compromise the effectiveness of all attack baselines. One important reason is that FT-SAM adopts Sharpness-Aware Minimization [12] to adjust the outlier of weight norm (large norms) to remove the potential backdoor. Larger weights of neurons are introduced by existing attacks to guarantee a high ASR [30], which also causes large differences when receiving benign and backdoor inputs (see Figure 11.4). `Grond` can bypass FT-SAM, as expected, since it deliberately decreases the weights of backdoor neurons, compromising the core working mechanism of FT-SAM.

Comparison with Supply-Chain Attacks. Sharing a similar threat model to supply-chain attacks, we compare `Grond` and three state-of-the-art supply-chain attacks (SSDT [36], DFST [7], and DFBA [2]), where these attacks are also designed to be robust against backdoor defenses. In particular, DFST [7] proposes to include a controlled detoxification technique in the training process, which restrains the model from picking up simple features. DFBA [2] directly modifies a few parameters of a classifier to inject a backdoor. SSDT [36] introduces additional terms in the loss for the Source-Specific and Dynamic-Triggers (i.e., SSDT) attack, which obscures the difference between normal samples and

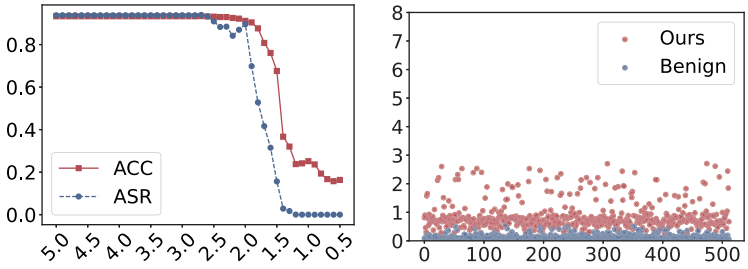


Figure 11.4: The left is the performance of pruning neurons with high TAC values using different thresholds for `Grond`. The right is the TAC analysis of `Grond` on 512 neurons.

malicious samples. Tables 11.2 and 11.3 also include the performance of supply-chain attacks against pruning- and fine-tuning-based defenses. It is clear that existing backdoor defenses can defeat supply-chain attacks. The ASR of DFST [7], DFBA [2], and SSDT [36] are decreased to less than 10% while the BA drop is less than 3%.

On ImageNet200 and GTSRB. Real-world classification tasks may involve more categories, such as GTSRB (43 classes) and ImageNet200 (200 classes), and the percentage of each class in the dataset will commonly be much less than 10%. We target InceptionNext-Small on Imagenet200 and ResNet18 on GTSRB. The l_∞ norm perturbation budget of UPGD is $\epsilon = 16$ for GTSRB and $\epsilon = 8$ for ImageNet200 to achieve imperceptible perturbations. Table 11.4 demonstrates that `Grond` is still effective on datasets with more classes and higher resolutions, especially against the most powerful parameter-space defense, FT-SAM.

11.4.3. ADAPTIVE DEFENSES

As `Grond` includes UPGD trigger and Adversarial Backdoor Injection (ABI), we consider two adaptive defenses targeting those two components.

TAC pruning targeting the UPGD trigger. First, we use the UPGD trigger information to build a new pruning method based on the TAC values. The TAC values are calculated using the backdoor trigger, making the TAC highly adaptive to evaluating any backdoor attack. In particular, we prune neurons with high TAC values in the backdoored model, i.e., removing the neurons more sensitive to the UPGD trigger. Figure 11.4 shows the pruning results of `Grond`. The left figure provides the pruning results. The right figure contains the TAC values plots of neurons in the 4_{th} layer (the layer before the classification head) of ResNet18. We show that pruning neurons with high TAC values decreases benign accuracy, which means the backdoor neurons are not easily distinguishable from benign neurons without harming benign performance. The analysis supports our statement that `Grond` spreads the backdoor to more neurons instead of a few prominent ones. In Section 11.5.6, we provide sorted TAC value plots in Figure 11.11, showing that prominent neurons with high TAC values are rather limited in `Grond`.

Neuron noise targeting ABI. Second, we consider adding noise to neurons of `Grond` models as an adaptive defense, as the ABI component involves operations on neuron weights. Specifically, we add noise to the weights and biases of batch normalization layers.

Table 11.8: Adaptive defense against Grond using noise on neurons.

Method	Benign		Grond	
	BA	ASR	BA	ASR
$\epsilon_{noise} = 0.0$	94.76	-	94.16	98.04
$\epsilon_{noise} = 0.1$	94.54	-	93.88	97.99
$\epsilon_{noise} = 0.2$	93.82	-	93.18	96.22
$\epsilon_{noise} = 0.3$	91.21	-	90.05	99.08
$\epsilon_{noise} = 0.4$	85.65	-	88.33	99.42
$\epsilon_{noise} = 0.5$	82.43	-	84.30	99.74

The range of noise is limited by $[-\epsilon_{noise}, \epsilon_{noise}]$. Table 11.8 provides the performance of Grond and the benign model under different noise levels. With increased noise, the benign accuracy of both benign and Grond models is decreased, but the ASR of Grond remains high. This result also supports the stealthiness of Grond in the parameter space.

11.4.4. BACKDOOR ANALYSIS

This section analyzes why baseline attacks are ineffective against parameter space backdoor defenses, and why Grond performs better, according to the TAC values and the weights' change after applying backdoor defenses.

Pruning neurons with prominent TAC values for baseline attacks. Section 11.3.2 demonstrates the existence of prominent neurons with high TAC values in backdoor models. Note that TAC represents the strongest type of backdoor defense, where the exact trigger information is exploited. In this section, we show that pruning prominent neurons could mitigate all baseline attacks. Specifically, we assign zero to the neuron's weight if its TAC value exceeds a certain threshold. Figure 11.3 shows the pruning performance using different thresholds for ten baseline attacks. We can observe that for all ten baseline attacks, one threshold can always be found where the ACC is high and the ASR is low. It indicates that the backdoor effect of the ten attacks can be erased while maintaining good performance on benign samples if these prominent neurons are pruned. Thus, we confirm the existence of prominent neurons, which correspond to the backdoor effect. More importantly, these backdoor neurons can be disentangled from benign neurons, so these baseline attacks are not stealthy concerning neuron weights, i.e., not stealthy in the parameter space.

Grond's weights are more difficult to be modified by backdoor defense. This section analyzes the changes while applying pruning- and fine-tuning-based defenses to baseline attacks and Grond. Our goal is to demonstrate that the baseline attacks can be significantly affected by defenses, but Grond can resist these defenses. In Figure 11.5, we record the changes in the weights of 512 neurons (in layer 4 of ResNet18) after applying two types of defenses, FT-SAM [73] and CLP [71], as they both focus on the backdoor-related neurons. In contrast to BadNets, the weight changes in Grond model after FT-SAM fine-tuning are smaller. In the sorted changes, it is clear that there are a few neurons for BadNets that correspond to significant changes, which is not the case for Grond. In the second row of

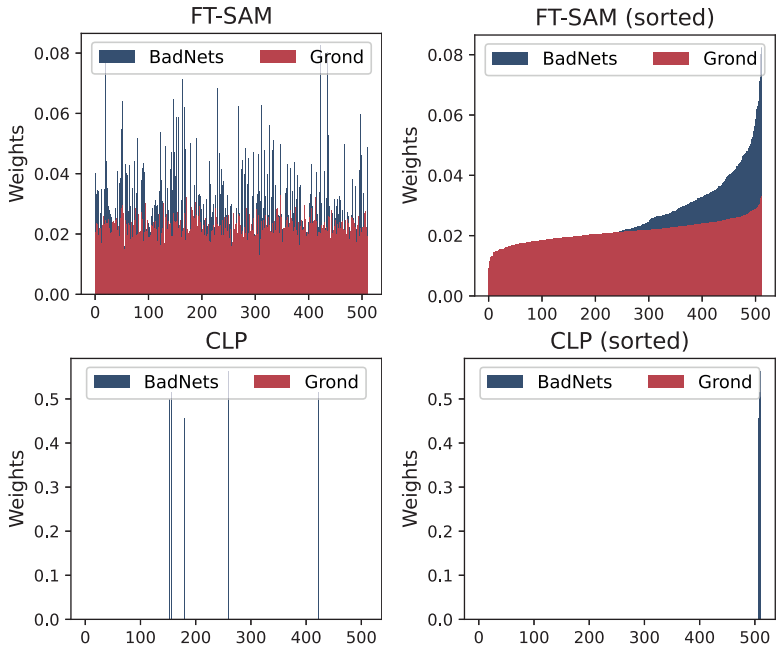


Figure 11.5: The weight changes after backdoor defenses. More results with other attacks in Figure 11.12.

Figure 11.5, the CLP can find a few neurons relevant to backdoor but cannot find these for `Grond`. In addition, pruning-based methods (as shown by CLP results in Figure 11.5) only improve a few neurons, while fine-tuning methods can update all neurons. We conjecture this is why fine-tuning-based defenses perform better than pruning-based defenses in Tables 11.2 and 11.3. More results with other baseline attacks are provided in Figure 11.12.

11.4.5. ABI IMPROVES COMMON BACKDOOR ATTACKS

In this section, we show that our Adversarial Backdoor Injection (ABI) strategy generalizes to all evaluated common backdoor attacks. We combine the ABI module with baseline attacks to improve their resistance against parameter-space defenses. Figure 11.6 demonstrates that ABI is effective for all attacks when evaluating against the parameter-space defense ANP, where ASRs increase after adversarial injection, especially for BadNets, Blend, AdvDoor, Narcissus, and Adap-Blend. The improvement for feature space attacks (WaNet, IAD, and Bpp) is incremental. We speculate that feature space attacks rely too much on prominent features, as their modification in the input space is minor. To activate the backdoor with such minor input modifications, the prominent features are required in the feature space. In addition, Figure 11.10 shows the results of ABI without defense, demonstrating that it does not harm in general the BA and ASR when no defense is applied. Following our findings, we suggest that future backdoor attacks can use ABI to increase parameter-space stealthiness.

Table 11.9: Backdoor detection performance on CIFAR10. 20 ResNet18 models are trained at each poisoning rate. Bd. refers to the number of models determined as backdoor models. Acc. refers to the detection accuracy.

Defense	PR=5%		PR=1%		PR=0.5%	
	Bd.	Acc.	Bd.	Acc.	Bd.	Acc.
NC [56]	5	25%	2	10%	1	5%
Tabor [16]	5	25%	2	10%	0	0%
FeatureRE [58]	0	0%	0	0%	0	0%
Unicorn [59]	0	0%	0	0%	0	0%
BTI-DBF [65]	3	15%	5	25%	3	15%

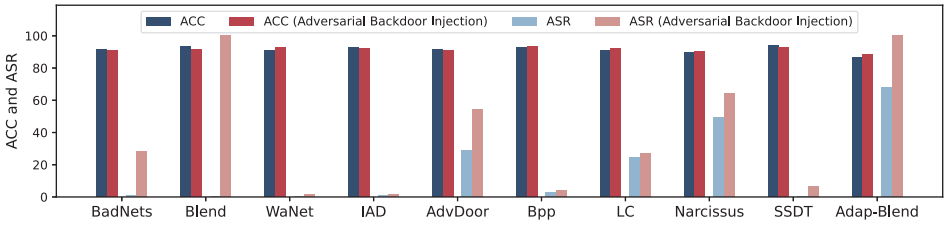


Figure 11.6: BA and ASR of backdoor attacks before and after ABI against parameter-space defense ANP.

11.4.6. BACKDOOR DETECTION

Following previous works [65, 64], we choose five representative backdoor model detections for evaluation. We use 20 models for each poisoning rate with different random seeds. Then, we report the number of models detected as backdoor models out of the 20. Table 11.9 shows that all detections fall short when detecting `Grond`. In particular, NC [56], Tabor [56], and BTI-DBF [65] can detect a small part of backdoored models, while FeatureRE [58] and Unicorn cannot detect any of them. For featureRE [58], we conjecture it is over-dependent on the separability in the feature space, but `Grond` does not rely on prominent backdoor features according to Figure 11.9. For Unicorn [59], the false positive rate is high, and it tends to report every class as the backdoor target, even on models trained with benign data only. Except for model detection, `Grond` can also bypass input-space detections as demonstrated in Section 11.5.2.

Table 11.10: Comparison with different strategies for the generation of backdoor triggers.

Strategy	No Defense		CLP [71]		FT-SAM [73]	
	BA	ASR	BA	ASR	BA	ASR
Random noise	94.24	1.28	94.13	0.97	93.90	1.84
PGD	94.77	69.33	92.57	46.63	92.40	24.56
UPGD	93.43	98.04	93.29	87.89	92.02	80.07

Table 11.11: The semantic trigger (an automobile image) with different strategies for `Grond`. The "Target Same" refers to the target class being the same as the semantic trigger, i.e., automobile. The "Target Diff." refers to the target class being different (i.e., airplane) from the semantic trigger.

Strategies for 		No Defense		CLP [71]		FT-SAM [73]	
		BA	ASR	BA	ASR	BA	ASR
Clean-Label	Target Same	94.20	74.64	93.97	70.66	91.68	9.38
	Target Diff.	93.82	94.42	93.75	93.50	90.92	31.37
Dirty-Label	Target Same	93.77	100	93.88	100	91.40	12.17
	Target Diff.	94.25	100	94.14	100	91.34	7.39

Table 11.12: Ablation study for `Grond`.

Arch	Method	No Defense		CLP [71]		FT-SAM [73]	
		BA	ASR	BA	ASR	BA	ASR
ResNet18	UPGD	93.86	98.61	91.15	3.97	91.80	51.77
	+ABI	93.43	98.04	93.29	87.89	92.02	80.07
InceptionNeXt	UPGD	87.81	96.81	87.72	96.57	87.06	2.37
	+ABI	87.06	96.86	86.93	96.87	86.50	92.02

11.4.7. ABLATION STUDY

The ablation study is designed for the two components of `Grond`, the UPGD trigger and Adversarial Backdoor Injection. In addition, we also evaluate the dirty-label setting of `Grond` and show the difference compared to using clean-label.

Trigger generation. To explore the influence of trigger patterns, we employ and evaluate three types of triggers: random noise, PGD perturbation, and UPGD perturbation, using ResNet18 on CIFAR10. The random noise is sampled from a uniform distribution, and the PGD employs a projected gradient descent to generate sample-wise perturbations [34]. The generation of UPGD is described in Algorithm 4. All three triggers are limited to 8/255 (l_∞ norm) for imperceptibility and use the same training settings described in Table 11.5.

Table 11.10 shows that random noise is ineffective as a backdoor trigger due to low

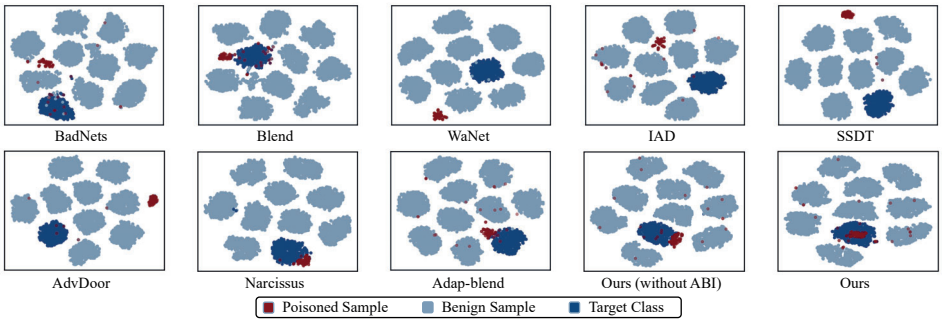


Figure 11.7: Examples of feature visualization of `Grond` and baseline attacks.

Table 11.13: Evaluation with the proactive defense, CT [42], under different poisoning rates (PR).

Attack	PR	ACC	ASR	Recall	FPR
BadNets [14]	5%	93.18	99.96	2500/2500	1568/47500
	2.5%	93.35	99.83	1250/1250	518/48750
	1%	93.30	100	500/500	73/49500
	0.5%	93.43	100	250/250	5/49750
	0.3%	93.63	99.94	150/150	222/49850
Adap-patch [41]	5%	93.28	100	1808/2500	116/47500
	2.5%	93.68	100	1088/1250	20/48750
	1%	93.73	100	494/500	570/49500
	0.5%	93.31	100	160/250	154/49750
	0.3%	93.26	100	86/150	3825/49850
`Grond`	5%	93.84	99.41	2499/2500	671/47500
	2.5%	93.81	95.83	115/1250	7220/48750
	1%	94.09	92.48	208/500	6690/49500
	0.5%	94.36	92.91	90/250	6738/49750
	0.3%	94.22	90.10	29/150	6349/49850

ASR, even if no defense is applied. The sample-wise PGD perturbation is more effective than random noise and shows (limited) robustness against CLP and FT-SAM. UPGD generates the most effective backdoor trigger with an ASR higher than 80% after CLP and FT-SAM, and we speculate that the reason is that UPGD exploits features from the target class, similar to Narcissus [68].

Concerning exploiting features from the target class, we also explore using the natural image as the trigger, which directly contains the semantic information. Table 11.11 shows the performance when using an automobile image from the CIFAR10 dataset as a semantic trigger. Specifically, inspired by naturally occurring backdoor [21], we design the semantic trigger by resizing the automobile image to 8×8 and sticking it on a part of (PR=5%) the training images. We consider four types of strategies, including only poisoning the same class as the trigger in clean-label, poisoning a different class (class airplane) in clean-label, only poisoning the same class as the trigger in dirty-label, and poisoning a different class (class airplane) in dirty-label. In Table 11.11, the semantic trigger can be effective as a backdoor trigger and resist against the CLP pruning. However, the semantic trigger is not robust against the FT-SAM fine-tuning. The reason is that the semantic trigger cannot effectively represent the feature of a class. Conversely, adversarial perturbation (PGD, UPGD) acquires more representative and discriminative class features, since they can capture the correlations among different regions of the decision boundary and easily fool most of the inputs [37].

Adversarial backdoor injection is critical. There are two components in `Grond`: the UPGD trigger generation and Adversarial Backdoor Injection. We conduct an ablation study with two architectures on CIFAR10 to analyze the impact of the ABI component. As shown in Table 11.12, after removing the ABI component, CLP or FT-SAM can defend against the clean-label attack with the UPGD trigger. Thus, the Adversarial Backdoor Injection is the key component in maintaining the effectiveness of backdoor attacks against parameter-space defenses.

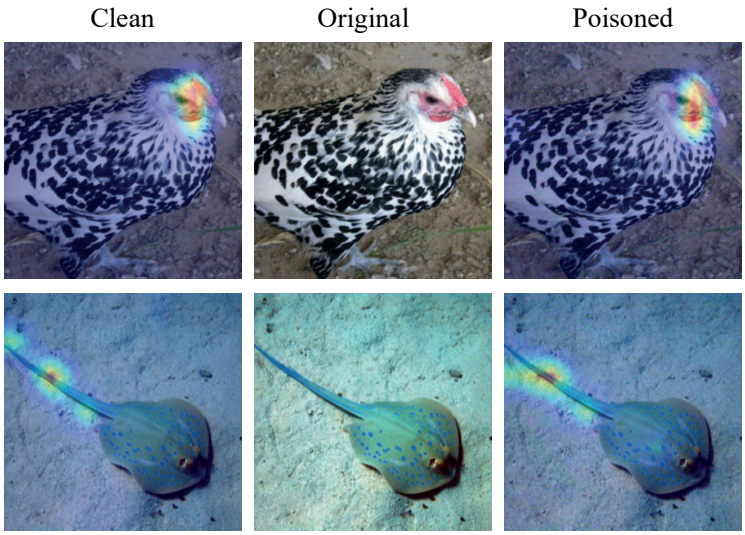


Figure 11.8: Examples of Grad-CAM activation map with ImageNet200 images by clean and `Grond` models. The first column is Grad-CAM maps with clean images, and the third column is Grad-CAM maps with `Grond`-poisoned images.

Dirty-label `Grond`: `Grond` works well in a clean-label setting that uses an invisible trigger and does not change the original labels of the poisoned samples. However, as we use the supply-chain threat model (the attacker has access to the training process), we could also explore the effect of a dirty-label backdoor attack. A dirty-label threat model could simplify the backdoor by poisoning samples from any class, while the clean-label is limited to a single class.

Table 11.14 considers two types of dirty-label `Grond`. The "One Target" uses the same setting and trigger as clean-label `Grond`, i.e., misclassifying any inputs with the trigger to the one target class. The "Dynamic Target" refers to misclassifying inputs from different classes to different targets. In addition, "Dynamic Target" `Grond` uses non-target UPGD as the trigger, which is generated by maximizing the loss between $\boldsymbol{\theta} x + \boldsymbol{\delta}$ and the true label.

According to the results, One Target `Grond` achieves very high ASR. Dynamic Target `Grond` performs worse than One Target since Dynamic Target is a more complex task. Overall, dirty-label `Grond` can still perform well, and even has a higher ASR than the clean-label setting. However, dirty-label `Grond` is less robust than clean-label `Grond` against backdoor defenses because it is less stealthy. For example, FT-SAM can decrease the ASR of Dynamic Target `Grond` (PR=5%) from 94.22% to below 10%, and decrease One Target ASR from 100% to 0.0%. We conjecture this is because the dirty-label setting obfuscates the benign semantics of images to their true labels, as the poisoned samples in each class have been assigned the backdoor target label. In contrast, only the target class is poisoned in a clean-label setting.

Table 11.14: Evaluation of dirty-label Grond using ResNet18 on CIFAR10. The dynamic target refers to samples from different classes having different backdoor targets.

Dirty-Label Grond	PR	No Defense		CLP [71]		FT-SAM [73]	
		BA	ASR	BA	ASR	BA	ASR
One Target	5%	91.60	100	91.41	100	90.24	0.00
	1%	93.64	100	91.13	40.83	91.20	46.97
	0.5%	94.35	100	91.84	97.03	91.77	99.04
Dynamic Target	10%	94.26	91.95	94.13	77.29	91.67	1.46
	5%	94.22	90.43	94.04	90.34	91.45	5.48
	1%	94.33	81.82	94.26	81.80	92.36	15.77

11.4.8. DIFFERENT ARCHITECTURES WITH DIFFERENT SURROGATE MODELS

We evaluate Grond with four additional victim architectures in Table 11.16: VGG16, DenseNet121, EfficientNet-B0, and InceptionNeXt-Tiny. In addition, as Grond requires a surrogate model to generate UPGD as the backdoor trigger, we provide the results when UPGD is generated using different architectures for the surrogate model. For each architecture, UPGD is generated by either the victim architecture or ResNet18 to perform our attack. In Table 11.16, we use the three most powerful defenses according to Tables 11.2 and 11.3. Regardless of the model's architectures or the architectures for UPGD, Grond bypasses most defenses. This is because the UPGD contains semantic information of the target class and can be transferred among different architectures [37]. In a few cases, using UPGD generated by the same architecture shows better attack performance. For example, conducting Grond on InceptionNeXt-Tiny with UPGD generated by InceptionNeXt-Tiny shows ASRs above 90%, but also a much lower ASR when using UPGD generated by ResNet18. We conjecture that transferring UPGD from ResNet18 to InceptionNeXt-Tiny is more difficult than transferring it to other architectures due to the large convolution kernel design of InceptionNeXt.

11.4.9. VISUALIZATION

Grad-CAM cannot spot the trigger area of Grond. Grad-CAM [47] was originally designed to visualize the network's preference when taking an input image. In backdoor defense research, Grad-CAM is leveraged to highlight the important areas in order to detect the potential backdoor trigger area [6]. Figure 11.8 shows the activated area of a clean model and Grond backdoored model using Grad-CAM. The activated area of Grond backdoored model is indistinguishable from the clean model, so the Grad-CAM-based defense [6] is also ineffective against Grond.

t-SNE visualization of feature space. Figure 11.7 shows the latent feature (feature space of the last convolutional layers) from Grond backdoor models with and without adversarial backdoor injection in 2-D space and other baseline attacks by t-SNE [55]. The poisoning rate for all is 0.5%. WaNet cannot achieve satisfactory ASR at this very low poisoning rate, so we use the default setting according to their open-source implementation. Specifically,

we perform dimensionality reduction for the latent features by t-SNE. The model architecture is ResNet18 and trained on CIFAR10. Each class of samples forms a tight cluster, and Grond poisoned samples are better mixed with the target class samples when the model is trained with adversarial backdoor injection.

11.5. STRONGER DEFENDERS AND ADDITIONAL ANALYSIS

11.5.1. PROACTIVE DEFENSE

Real-world powerful defenders could take more initiative by intervening proactively in the attack process and exploiting poisoned data. We evaluate Grond against the SOTA proactive defense, CT [42], that detects poisoned samples in the training data. Specifically, CT considers data from the original poisoned training data as regular batches and introduces randomly labeled benign data as confusing batches. Then, CT performs normal supervised training on both regular and confusing batches to produce an inference model, aiming to corrupt benign semantic features and correlations with correct labels in the inference model by confusing batches. The backdoor effect remains in the inference model because there is no trigger information in the confusing batches, and correctly predicted samples by the inference model are recorded as poisoned.

Table 11.13 presents the detection results on two baseline attacks (BadNets [14] and Adap-patch [41]) and Grond. CT is effective against the two baseline attacks, where most poisoned samples in the training set are detected with a relatively low false positive rate. However, CT is not capable of detecting Grond when the poisoning rate is lower than 5% due to a high false positive rate and low recall. To understand why CT is not effective against Grond, we recall that the main idea of CT is to corrupt benign semantic features and their correct label, but not corrupt backdoor semantic features. However, Grond utilizes the benign semantic features of the target class to generate UPGD perturbation as the trigger. CT's mechanism also corrupts the backdoor features of Grond. Therefore, CT cannot effectively detect Grond poisoned samples.

Table 11.15: Input-space detection results.

Attack	Scale-up [15]				IBD-PSC [19]			
	TPR	FPR	AUC	F1	TPR	FPR	AUC	F1
BadNets [14]	81.93	32.90	0.7627	0.7524	100	7.90	0.9996	0.9606
Blend [5]	99.32	38.74	0.8681	0.8275	100	0.90	1.00	0.9953
Adap-Blend [41]	68.72	18.99	0.7621	0.7297	53.95	11.77	0.8731	0.6495
Grond (PR=5%)	24.40	17.69	0.5463	0.3409	0.00	10.33	0.5698	0.0
Grond (PR=1%)	18.39	17.96	0.4879	0.2656	0.00	5.82	0.0626	0.0
Grond (PR=0.5%)	7.05	16.19	0.4034	0.1113	0.00	4.82	0.1087	0.0

11.5.2. DETECTION OF BACKDOOR INPUT.

Backdoor input detection is a defense technique that determines whether or not a given input includes a backdoor trigger. We show that Grond-generated backdoor samples can resist established backdoor detection methods. Table 11.15 shows the input-space

Table 11.16: Grond against defenses using different architectures on CIFAR10 with a poisoning rate of 5%. The surrogate indicates the architecture used to generate UPGD as the trigger.

Victim	Surrogate	No Defense		FT-SAM [73]		I-BAU [67]		FST [35]	
		BA	ASR	BA	ASR	BA	ASR	BA	ASR
VGG16	ResNet18	92.69	95.31	92.72	78.42	90.10	14.53	89.12	92.68
	VGG16	92.57	90.10	92.22	95.14	90.20	76.51	91.72	90.58
DenseNet121	ResNet18	92.39	95.62	90.98	23.88	86.73	48.14	90.77	88.94
	DenseNet121	92.38	81.07	91.10	16.91	90.90	54.76	91.13	71.29
EfficienNet-B0	ResNet18	87.7	96.23	84.05	71.07	87.64	95.41	82.07	97.67
	EfficienNet-B0	86.92	92.61	83.77	71.17	86.93	92.13	82.45	68.72
InceptionNeXt-Tiny	ResNet18	85.07	91.83	85.07	2.17	85.25	91.67	82.78	3.82
	InceptionNeXt-Tiny	85.54	96.24	85.64	90.14	85.49	97.21	83.92	97.29

detection results using Scale-up [15] and IBD-PSC [19]. We report the True Positive Rate (TPR), False Positive Rate (FPR), AUC, and F1 score in Table 11.15 for baseline attacks and Grond, where Scale-up and IBD-PSC are effective against three baseline attacks but cannot detect Grond-generated backdoor samples.

11.5.3. LARGE AND TRANSFORMER-BASED ARCHITECTURES

To further evaluate the effectiveness of Grond on large-sized and especially transformer-based models, we conduct experiments on various architectures, including ResNet101 [17], Swin-Base [32], Swin-Large [32], ViT-Base [11], and ViT-Large [11].

There are two components (UPGD trigger and ABI) of Grond that are necessary to be applied to transformer-based architectures. The UPGD trigger generation algorithm is compatible with transformer-based architectures, as transformers' gradient information can be used to generate UPGD perturbations. However, UCLC [71] for ABI is incompatible with the attention mechanism. The UCLC is designed for feature channels of convolutional layers, which do not exist in the attention mechanism. Therefore, we design a tailored novel pruning method against backdoor neurons in the attention-based models. Specifically, we consider an attention head as a "neuron", the new pruning method then calculates the singular value decomposition of each attention head as an index of relevance to the backdoor effect. The process can also be considered as solving for the approximation of the Lipschitz Constant for each attention head according to [71]. Then, the sensitive heads are updated using the following equation:

$$\boldsymbol{\theta}_l^{(k)} := \begin{cases} \boldsymbol{\theta}_l^{(k)} \times 0.5, & \sigma(\boldsymbol{\theta}_l^{(k)}) > \text{mean}(\sigma(\boldsymbol{\theta}_l)) + u \times \text{std}(\sigma(\boldsymbol{\theta}_l)) \\ \boldsymbol{\theta}_l^{(k)}, & \text{otherwise,} \end{cases} \tag{11.4}$$

where $\boldsymbol{\theta}_l^{(k)}$ is the weight of k_{th} head at l_{th} layer, σ is the maximum of values in the singular value decomposition matrix of $\boldsymbol{\theta}_l^{(k)}$. We evaluate the performance of Grond on transformer architectures with three fine-tuning-based defenses. Table 11.17 provides the experimental results and demonstrates the effectiveness of Grond on large and transformer architectures. Notably, the three fine-tuning defenses cannot effectively decrease the ASR of Grond models. We conjecture this is because injecting a stealthy backdoor into

larger models is easier.

The hyperparameters for training ResNet101 are the same as those in Table 11.5. For transformer models, the number of training epochs is 100 for both CIFAR10 and ImageNet200. The perturbation budget of UPGD for transformer models is $\epsilon = 16$. The hyperparameters not mentioned are given in Table 11.5.

Table 11.17: The performance of `Grond` on large and transformer-based architectures.

Dataset	Victim	Surrogate	Parameters (M)	No Defense		vanilla FT		FT-SAM [73]		FST [35]	
				BA	ASR	BA	ASR	BA	ASR	BA	ASR
CIFAR10	ResNet101	ResNet18	42.51	93.75	99.22	87.37	99.92	92.78	99.04	93.65	99.77
	ViT-Base	ViT-Base	85.27	80.24	98.80	76.43	98.72	79.50	98.10	79.77	98.73
	ViT-Large		302.60	78.92	98.89	74.47	97.53	77.98	98.56	78.53	98.53
	Swin-Base	Swin-Base	86.70	83.08	99.65	81.61	99.80	80.21	98.88	82.44	99.86
	Swin-Large		194.93	83.04	99.69	80.79	99.35	80.99	99.47	79.31	99.53
ImageNet200	ViT-Base	ViT-Base	85.95	83.41	98.36	82.08	98.86	83.48	98.24	80.99	97.84
	ViT-Large		303.51	86.08	89.46	83.19	89.07	86.06	90.52	82.36	85.35
	Swin-Base	Swin-Base	86.95	74.68	97.82	72.56	98.29	75.75	96.11	72.29	94.77
	Swin-Large		195.30	76.99	95.36	74.29	97.91	75.35	90.77	73.07	90.72

11.5.4. BACKDOOR ANALYSIS BY FEATURE SPACE INVERSION

We also provide a feature space analysis for different attacks by using a feature mask to decouple the benign and backdoor features following BTI-DBF [65] and BAN [64]. The decoupling assumes that benign features related to the correct prediction introduce a lower loss, and the backdoor features related to backdoor prediction introduce a higher loss. In particular, the benign and inversed backdoor features are decoupled as follows:

$$\min_{\boldsymbol{\theta} m} \sum_{(\boldsymbol{x},y)\in\mathscr{D}_l} \Big[\mathscr{L}\big(f_L \circ (g(\boldsymbol{x}) \odot \boldsymbol{m}), y\big) - \mathscr{L}\big(f_L \circ (g(\boldsymbol{x}) \odot (1-\boldsymbol{m}), y)\big) + \lambda|\boldsymbol{m}|\Big], \tag{11.5}$$

where $g = f_{L-1} \circ \cdots f_1$ is the model without the classification head. $\boldsymbol{\theta} m$ is the learned feature mask, and $\mathscr{D}_l$ is a small set of benign samples with correct labels. As validated with BTI-DBF [65] and BAN [64], benign and backdoor features can be decoupled by the mask $\boldsymbol{\theta} m$, after which backdoor features will introduce a substantially higher loss with respect to the ground truth label.

Decoupled benign feature loss and backdoor feature loss of all evaluated attacks are demonstrated in Figure 11.9, where benign feature loss is represented by the first term in Equation (11.5) and backdoor feature loss by the second term. It can be observed that several backdoor attacks introduce prominent backdoor features, such as WaNet, IAD, and Bpp, which result in substantially higher backdoor feature loss than benign feature loss. In contrast, several backdoor attacks introduce less prominent backdoor features, including AdvDoor, Narcissus, Adap-blend, and `Grond`. Revisiting Tables 11.2 to 11.4, attacks that introduce prominent backdoor features are more susceptible to backdoor defenses than attacks with less prominent backdoor features. Both TAC and feature space inversion analyses further confirm that `Grond` provides comprehensive stealthiness.

Hyperparameters for the Inversed Backdoor Feature Loss Following the settings in BTI-DBF [65] and BAN [64], we use Adam and the learning rate of 0.01 to search for 20

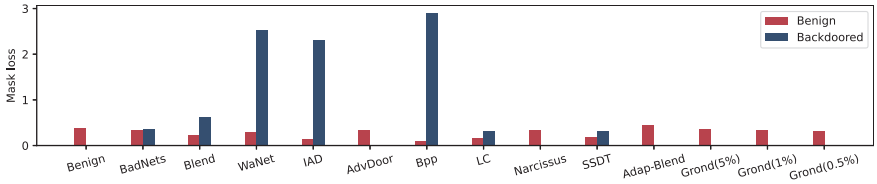


Figure 11.9: Benign and inversed backdoor feature loss (Equation (11.5)) for all baseline attacks. Large backdoored loss indicates that the backdoor is prominent in the feature space.

Table 11.18: Evaluation of `Grond` against adversarial training (AT) on CIFAR10. RA refers to accuracy under PGD [34] adversarial attack.

Arch	No Defense			AT		
	BA	ASR	RA	BA	ASR	RA
ResNet18	93.43	98.04	3.82	88.56	89.69	62.24
VGG16	92.57	90.10	9.64	85.64	85.78	59.75
DenseNet121	92.38	81.07	9.72	86.67	64.30	58.66

epochs for the feature mask in Equation (11.5). The optimization of the mask uses 1% of the training data. λ is 0.72. The elements in the mask are limited to continuous values between 0 and 1.

11.5.5. ADVERSARIAL BACKDOOR INJECTION DOES NOT IMPACT BACKDOOR EFFECTIVENESS IN CASE OF NO DEFENSE.

Figure 11.10 shows additional Adversarial Backdoor Injection results against models without defense. We show that Adversarial Backdoor Injection does not influence the backdoor effectiveness in general when no defense is applied.

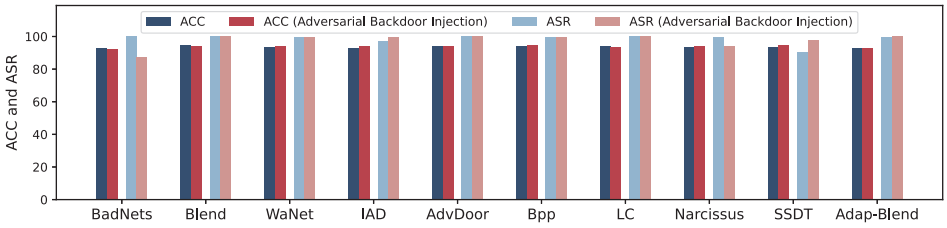


Figure 11.10: The attack performance (no defense) when combined with Adversarial Backdoor Injection.

11.5.6. FURTHER TAC ANALYSIS

For a clearer demonstration, we also provide sorted TAC value plots in Figure 11.11, which sort the TAC values in Figure 11.2. Figure 11.11 demonstrates the existence of prominent neurons, and `Grond` is more stealthy.

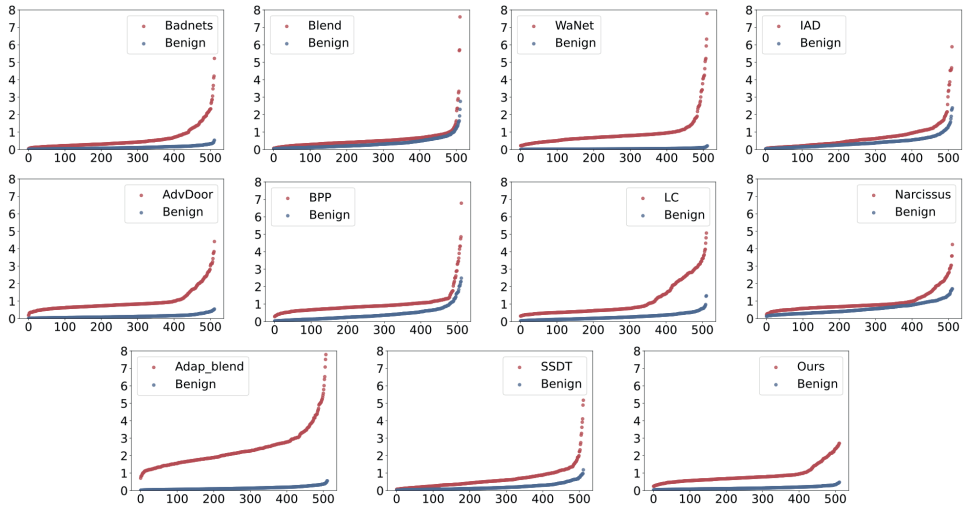


Figure 11.11: TAC plots of sorted TAC values, which show the prominent neurons of baseline attacks. The y axis contains the TAC values, and the x axis is the index of neurons. Prominent neurons are not found in our attack.

11.5.7. ADVERSARIAL TRAINING AS A DEFENSE

We evaluate `Grond` with adversarial training (AT), which is considered one of the most successful defenses against adversarial attacks, since the UPGD trigger is a special type of adversarial perturbation. Table 11.18 shows the backdoor performance and robustness of `Grond` before and after adversarial training using CIFAR10 with three architectures (ResNet18, VGG16, and DenseNet121). The robust accuracy against PGD attacks significantly increased after AT, confirming AT's effectiveness. However, AT does not decrease ASR, which means `Grond` backdoor can resist AT.

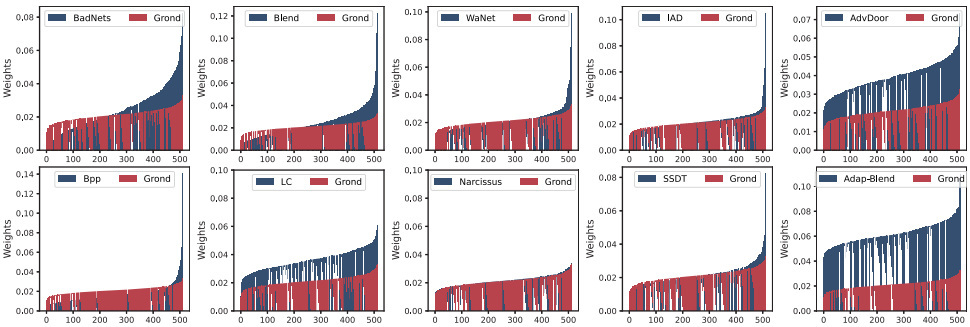


Figure 11.12: The weight changes after FT-SAM [73] for baseline attacks.

Table 11.19: Crafting inputs to minimize changes in the parameter space.

λ	No Defense		CLP	
	BA	ASR	BA	ASR
0.5	94.09	1.43	93.39	1.43
0.1	94.23	6.09	93.43	1.97
0.01	93.69	26.83	90.90	3.87
0.001	93.88	68.81	92.80	4.89
0.0005	94.13	90.78	92.81	5.84

11.5.8. WEIGHT CHANGES AFTER FT-SAM WITH 10 BASELINE ATTACKS

Figure 11.12 provides the results of weight changes after applying FT-SAM [73] to ten baseline attacks. The weight changes in baseline attacks show a significant difference in a few prominent neurons.

11.5.9. ATTEMPTS TO CRAFT INPUTS THAT MINIMIZE CHANGES IN THE PARAMETER SPACE

In this section, we also try to craft poisoned training data that minimizes changes in the parameter space instead of using ABI.

In the previous design of `Grond`, the backdoor is injected by the UPGD trigger. Then, `Grond`, through ABI, limits the changes of backdoor neurons by reducing their weights. We consider replacing the ABI component with an operation only in the training data, which does not require control over training. Our goal is to partially break the effectiveness of injecting the backdoor via the poisoned data. Specifically, we add poisoned data into the target class with a poisoning rate of $PR = 5\%$ under the clean-label setting, following the standard setting in previous work [59, 64]. Then, we add poisoned data into every class with a poisoning rate of $PR \times \lambda$. We aim to use the poisoned data in non-targeted classes to partially break the effectiveness of poisoned data in the targeted class. Table 11.19 provides the experimental results with different λ values. With the decrease in λ, both ASR (no defense) and ASR (after applying CLP) increase. This means that our attempt to minimize changes in the parameter space for stealthiness works. However, ASR after applying CLP is still lower than 10%, which means the attack is not robust against the backdoor defense.

11.6. EXAMPLES OF POISONED IMAGES

Figure 11.13 shows four training images from ImageNet200 when applied with UPGD. Please note that the images are only meant to demonstrate imperceptible trigger and poisoning perturbations. In our experiments, we only poison training images from the class “stingray” to inject the backdoor. The first row depicts poisoned images, while the second contains clean ones. Finally, the third row contains the residual of the first two rows. Notice that `Grond` does not introduce any visible difference to the clean images.

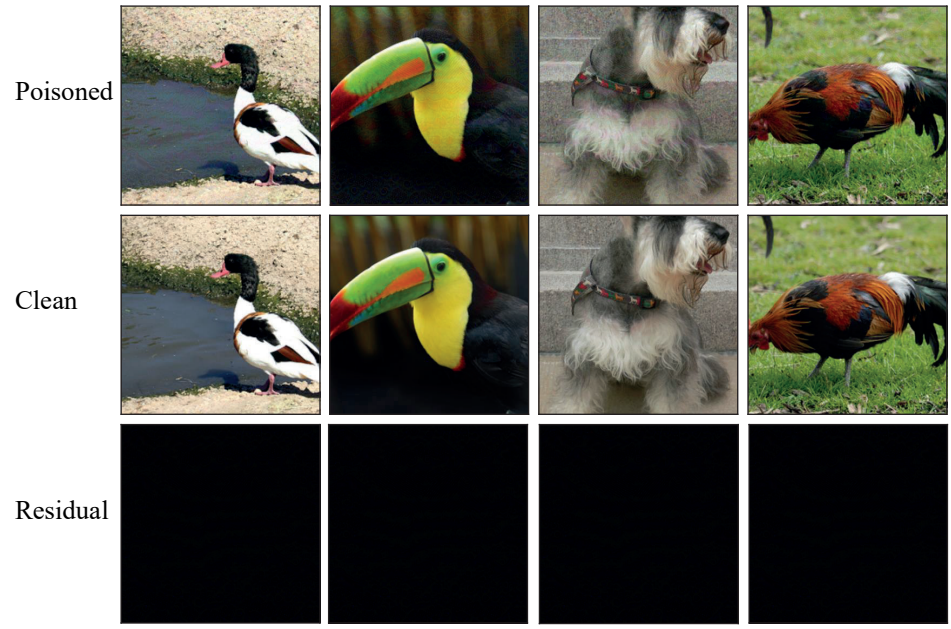


Figure 11.13: Examples of poisoned ImageNet200 images by `Grond`. We only poison training images from the class "stingray" in our experiments with ImageNet200.

11.7. CONCLUSIONS & FUTURE WORK

This paper studies whether backdoor attacks can resist diverse practical defenses and provides an affirmative answer: current common stealthy backdoor attacks are vulnerable to parameter-space defenses. We further explore how to increase the stealthiness of backdoor attacks against parameter-space defenses. We propose a novel supply-chain backdoor attack, `Grond`, that considers comprehensive stealthiness, including input, feature, and parameter-space stealthiness. `Grond` achieves state-of-the-art performance by leveraging adversarial examples and adaptively limiting the backdoored model's parameter changes during the backdoor injection to improve the stealthiness. We also show that `Grond`'s Adversarial Backdoor Injection can consistently improve other backdoor attacks against parameter space defenses. We suggest that future backdoor attacks should be evaluated against parameter-space defense. We also recommend that backdoor research explore Adversarial Backdoor Injection to enhance parameter-space stealthiness.

REFERENCES

[1] Eugene Bagdasaryan and Vitaly Shmatikov. "Blind Backdoors in Deep Learning Models". In: *USENIX Security*. 2021.

[2] Bochuan Cao et al. "Data Free Backdoor Attacks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2024.

[3] Bryant Chen et al. “Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering”. In: *SafeAI Workshop @ AAAI* (2018).

[4] Huili Chen et al. “ProFlip: Targeted Trojan Attack With Progressive Bit Flips”. In: *International Conference on Computer Vision (ICCV)*. 2021.

[5] Xinyun Chen et al. “Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning”. In: *arXiv* (2017).

[6] Hao Cheng et al. “Defending against backdoor attack on deep neural networks”. In: 2020.

[7] Siyuan Cheng et al. “Deep Feature Space Trojan Attack of Neural Networks by Controlled Detoxification”. In: 2021.

[8] Jia Deng et al. “ImageNet: A large-scale hierarchical image database”. In: *CVPR*. 2009.

[9] Khoa Doan, Yingjie Lao, and Ping Li. “Backdoor Attack with Imperceptible Input and Latent Modification”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[10] Khoa Doan et al. “LIRA: Learnable, Imperceptible and Robust Backdoor Attacks”. In: *International Conference on Computer Vision (ICCV)*. 2021.

[11] Alexey Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *International Conference on Learning Representations (ICLR)*. 2021.

[12] Pierre Foret et al. “Sharpness-aware Minimization for Efficiently Improving Generalization”. In: *International Conference on Learning Representations (ICLR)*. 2021.

[13] Yansong Gao et al. “STRIP: a defence against trojan attacks on deep neural networks”. In: *Annual Computer Security Applications Conference (ACSAC)*. 2019.

[14] Tianyu Gu et al. “BadNets: Evaluating Backdooring Attacks on Deep Neural Networks”. In: *IEEE Access* 7 (2019).

[15] Junfeng Guo et al. “SCALE-UP: An Efficient Black-box Input-level Backdoor Detection via Analyzing Scaled Prediction Consistency”. In: *ICLR*. 2023.

[16] Wenbo Guo et al. “Towards Inspecting and Eliminating Trojan Backdoors in Deep Neural Networks”. In: *ICDM*. 2020.

[17] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *CVPR*. 2016.

[18] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. “Handcrafted backdoors in deep neural networks”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 8068–8080.

[19] Linshan Hou et al. “IBD-PSC: Input-level Backdoor Detection via Parameter-oriented Scaling Consistency”. In: *International Conference on Machine Learning (ICML)*. 2024.

[20] Gao Huang et al. “Densely connected convolutional networks”. In: *CVPR*. 2017.

[21] Alaa Khaddaj et al. “Rethinking Backdoor Attacks”. In: *International Conference on Machine Learning (ICML)*. 2023.

[22] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).

[23] Yige Li et al. “Anti-Backdoor Learning: Training Clean Models on Poisoned Data”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[24] Yige Li et al. “Reconstructive Neuron Pruning for Backdoor Defense”. In: *International Conference on Machine Learning (ICML)*. 2023.

[25] Yiming Li et al. “BackdoorBox: A Python Toolbox for Backdoor Learning”. In: *International Conference on Learning Representations (ICLR) Workshop*. 2023.

[26] Junyu Lin et al. “Composite Backdoor Attack for Deep Neural Network by Mixing Existing Benign Features”. In: *ACM Conference on Computer and Communications Security (CCS)*. 2020.

[27] Weilin Lin et al. “Unveiling and Mitigating Backdoor Vulnerabilities based on Unlearning Weight Changes and Backdoor Activeness”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2024.

[28] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. “Fine-pruning: Defending against backdooring attacks on deep neural networks”. In: *International symposium on research in attacks, intrusions, and defenses*. Springer. 2018, pp. 273–294.

[29] Yannan Liu et al. “Fault injection attack on deep neural network”. In: *IEEE International Conference on Computer-Aided Design (ICCAD)*. 2017.

[30] Yingqi Liu et al. “ABS: Scanning Neural Networks for Back-doors by Artificial Brain Stimulation”. In: *CCS*. 2019.

[31] Yingqi Liu et al. “Trojaning Attack on Neural Networks”. In: *NDSS*. 2018.

[32] Ze Liu et al. “Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows”. In: *International Conference on Computer Vision (ICCV)*. 2021.

[33] Peizhuo Lv et al. “A Data-free Backdoor Injection Approach in Neural Networks”. In: *USENIX Security Symposium*. 2023.

[34] Aleksander Madry et al. “Towards Deep Learning Models Resistant to Adversarial Attacks”. In: *International Conference on Learning Representations (ICLR)*. 2018.

[35] Rui Min et al. “Towards Stable Backdoor Purification through Feature Shift Tuning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2023.

[36] Xiaoxing Mo et al. “Robust Backdoor Detection for Deep Learning via Topological Evolution Dynamics”. In: *SP*. 2024.

[37] Seyed-Mohsen Moosavi-Dezfooli et al. “Universal Adversarial Perturbations”. In: *IEEE / CVF Computer Vision and Pattern Recognition Conference (CVPR)*. 2017.

[38] Anh Nguyen and Anh Tran. “Input-Aware Dynamic Backdoor Attack”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[39] Tuan Anh Nguyen and Anh Tuan Tran. “WaNet - Imperceptible Warping-based Backdoor Attack”. In: *ICLR*. 2021.

[40] Ren Pang et al. “TrojanZoo: Towards Unified, Holistic, and Practical Evaluation of Neural Backdoors”. In: *Euro S&P*. 2022.

[41] Xiangyu Qi et al. “Revisiting the assumption of latent separability for backdoor defenses”. In: *ICLR*. 2023.

[42] Xiangyu Qi et al. “Towards A Proactive ML Approach for Detecting Backdoor Poison Samples”. In: *USENIX Security Symposium*. 2023.

[43] Xiangyu Qi et al. “Towards Practical Deployment-Stage Backdoor Attack on Deep Neural Networks”. In: *IEEE / CVF Computer Vision and Pattern Recognition Conference (CVPR)*. 2022.

[44] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. “TBT: Targeted Neural Network Attack With Bit Trojan”. In: *IEEE / CVF Computer Vision and Pattern Recognition Conference (CVPR)*. 2020.

[45] Adnan Siraj Rakin et al. “T-BFA: Targeted Bit-Flip Adversarial Weight Attack”. In: 2022.

[46] Yankun Ren, Longfei Li, and Jun Zhou. “Simtrojan: Stealthy backdoor attack”. In: *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2021, pp. 819–823.

[47] Ramprasaath R. Selvaraju et al. “Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization”. In: *International Conference on Computer Vision (ICCV)*. 2017.

[48] Reza Shokri et al. “Bypassing backdoor detection algorithms in deep learning”. In: *Euro S&P*. 2020.

[49] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *ICLR*. 2015.

[50] Johannes Stallkamp et al. “Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition”. In: *Neural Networks* (2012), pp. 323–332.

[51] Mingxing Tan and Quoc Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. In: *International Conference on Machine Learning (ICML)*. 2019.

[52] Di Tang et al. “Demon in the Variant: Statistical Analysis of DNNs for Robust Backdoor Contamination Detection”. In: *USENIX Security Symposium*. 2021.

[53] Dimitris Tsipras et al. “Robustness May Be at Odds with Accuracy”. In: *International Conference on Learning Representations (ICLR)*. 2019.

[54] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. “Label-consistent backdoor attacks”. In: *arXiv preprint arXiv:1912.02771* (2019).

[55] Laurens Van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE”. In: 2008.

[56] Bolun Wang et al. “Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks”. In: *SP*. 2019.

[57] Zhenting Wang, Juan Zhai, and Shiqing Ma. “BppAttack: Stealthy and Efficient Trojan Attacks Against Deep Neural Networks via Image Quantization and Contrastive Adversarial Learning”. In: *CVPR*. 2022.

[58] Zhenting Wang et al. “Rethinking the Reverse-engineering of Trojan Triggers”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022.

[59] Zhenting Wang et al. “UNICORN: A Unified Backdoor Trigger Inversion Framework”. In: *ICLR*. 2023.

[60] Shaokui Wei, Hongyuan Zha, and Baoyuan Wu. “Mitigating Backdoor Attack by Injecting Proactive Defensive Backdoor”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2024.

[61] Baoyuan Wu et al. “BackdoorBench: A Comprehensive Benchmark of Backdoor Learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022.

[62] Dongxian Wu and Yisen Wang. “Adversarial Neuron Pruning Purifies Backdoored Deep Models”. In: *NeurIPS*. 2021.

[63] Pengfei Xia et al. “Enhancing Backdoor Attacks With Multi-Level MMD Regularization”. In: 2023.

[64] Xiaoyun Xu et al. “BAN: Detecting Backdoors Activated by Adversarial Neuron Noise”. In: *arXiv preprint arXiv:2405.19928* (2024).

[65] Xiong Xu et al. “Towards Reliable and Efficient Backdoor Trigger Inversion via Decoupling Benign Features”. In: *ICLR*. 2024.

[66] Weihao Yu et al. “InceptionNeXt: When Inception Meets ConvNeXt”. In: *IEEE / CVF Computer Vision and Pattern Recognition Conference (CVPR)*. 2024.

[67] Yi Zeng et al. “Adversarial Unlearning of Backdoors via Implicit Hypergradient”. In: *International Conference on Learning Representations (ICLR)*. 2022.

[68] Yi Zeng et al. “Narcissus: A Practical Clean-Label Backdoor Attack with Limited Information”. In: *ACM Conference on Computer and Communications Security (CCS)*. 2023.

[69] Quan Zhang et al. “AdvDoor: adversarial backdoor attack of deep learning system”. In: *ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*. 2021.

[70] Zhendong Zhao et al. “DEFEAT: Deep Hidden Feature Backdoor Attacks by Imperceptible Perturbation and Latent Representation Constraints”. In: *IEEE / CVF Computer Vision and Pattern Recognition Conference (CVPR)*. 2022.

[71] Runkai Zheng et al. “Data-Free Backdoor Removal Based on Channel Lipschitzness”. In: *ECCV*. 2022.

[72] Nan Zhong, Zhenxing Qian, and Xinpeng Zhang. “Imperceptible Backdoor Attack: From Input Space to Feature Representation”. In: *International Joint Conferences on Artificial Intelligence (IJCAI)*. 2022.

[73] Mingli Zhu et al. “Enhancing Fine-Tuning Based Backdoor Defense with Sharpness-Aware Minimization”. In: *International Conference on Computer Vision (ICCV)*. 2023.

[74] Rui Zhu et al. “Gradient Shaping: Enhancing Backdoor Attack Against Reverse Engineering”. In: *Network and Distributed System Security (NDSS) Symposium*. 2024.

12

SoK: The Last Line of Defense: On Backdoor Defense Evaluation

Backdoor attacks pose a significant threat to deep learning models by implanting hidden vulnerabilities that can be activated by malicious inputs. While numerous defenses have been proposed to mitigate these attacks, the heterogeneous landscape of evaluation methodologies hinders fair comparison between defenses. This work presents a systematic (meta-)analysis of backdoor defenses through a comprehensive literature review and empirical evaluation. We analyzed 183 backdoor defense papers published between 2018 and 2025 across major AI and security venues, examining the properties and evaluation methodologies of these defenses.

Our analysis reveals significant inconsistencies in experimental setups, evaluation metrics, and threat model assumptions in the literature. Through extensive experiments involving three datasets (MNIST, CIFAR-100, ImageNet-1K), four model architectures (ResNet-18, VGG-19, ViT-B/16, DenseNet-121), 16 representative defenses, and five commonly used attacks, totaling over 3 000 experiments, we demonstrate that defense effectiveness varies substantially across different evaluation setups. We identify critical gaps in current evaluation practices, including insufficient reporting of computational overhead and behavior under benign conditions, bias in hyperparameter selection, and incomplete experimentation. Based on our findings, we provide concrete challenges and well-motivated recommendations to standardize and improve future defense evaluations. Our work aims to equip researchers and industry practitioners with actionable insights for developing, assessing, and deploying defenses to different systems.

This chapter is under review as G. Abad, K. Marina, S. Koffas, B. Tajalli, M. Arazzi, R. Riaño, X. Xu, Z. Liu, A. Nocera, and S. Picek, "SoK: The Last Line of Defense: On Backdoor Defense Evaluation." *arXiv preprint arXiv:2511.13143 (2025).*

12.1. Introduction

Deep learning (DL) has become a primary technology in critical application domains, from autonomous vehicles [35] to medical diagnosis and malware classification [7, 124]. As DL systems assume increasingly safety-critical roles, ensuring their security against adversarial manipulation has become increasingly important. Among the various threats to DL systems, including evasion attacks [140], model theft [60], and data poisoning [61], backdoor attacks have emerged as particularly insidious and difficult to detect.

Backdoor attacks implant hidden malicious behavior into models that is activated only when specific trigger patterns appear in the input [36]. Unlike evasion attacks that occur at inference time, backdoors are embedded during training and persist throughout the model's lifetime, making them exceptionally dangerous for deployed systems. The attack surface is broad: adversaries can poison training data [36, 15], manipulate the training process, or directly modify model parameters [177, 118]. Triggers can be embedded in the input space (e.g., pixel patterns [36]), the feature space (e.g., semantic perturbations [104, 22]), or the model weights, creating a diverse and evolving threat landscape.

The research community has responded with an extensive body of work on backdoor defenses, proposing detection and mitigation techniques that operate at different stages of the machine learning pipeline: pre-training (data sanitization), during training (robust learning), and post-training (model inspection) [80, 149, 89, 175]. This proliferation of defenses raises a critical question for practitioners: *Which defense should be deployed in production systems?* Answering this question requires a rigorous, fair comparison of defense effectiveness under realistic threat models. However, our analysis reveals that the current state of defense evaluation makes such comparisons nearly impossible.

The evaluation crisis. We conduct a systematic analysis of 183 backdoor defenses introduced in papers published in top-tier AI and security venues and listed in Table 12.1. Our investigation uncovers serious methodological issues that undermine confidence in reported defense performance:

- **Unrealistic threat models:** The vast majority of defenses are evaluated exclusively against outdated, dirty-label attacks (where attackers control training labels). Clean-label attacks, which preserve correct labels and represent more realistic threats, are rarely considered. Similarly, defenses are almost never tested against adaptive attackers who are aware of the defense mechanism.
- **Toy datasets dominate:** Over 73% of defenses are evaluated on MNIST or CIFAR-10. ImageNet-1K, a more challenging benchmark, appears in only ≈20% of papers. This raises serious concerns about generalizability to real-world deployment scenarios.
- **Limited trigger diversity:** Most evaluations use simple, visible patch-based triggers. Stealthy, imperceptible triggers are rarely considered, despite being more representative of sophisticated attacks.
- **Missing practical constraints:** Critical deployment considerations such as execution time, memory requirements, and computational cost are almost never reported, preventing practitioners from assessing real-world feasibility.

These evaluation gaps create a false sense of security: defenses appear highly effective on toy benchmarks but may fail under realistic threat models. When we re-evaluate

selected defenses on more challenging datasets and against adaptive, stealthy attacks, we observe substantial performance degradation that contradicts published claims.

Our contributions. This paper presents the first systematization of knowledge (SoK) focused exclusively on backdoor defense evaluation methodology. Rather than proposing new defenses or attacks, we analyze *how* the research community evaluates defenses and identify critical gaps that prevent meaningful comparison and deployment. Our work makes the following contributions:

- **Comprehensive literature analysis:** We systematically survey 183 backdoor defense papers from top-tier venues (2018–2025), cataloging their evaluation methodologies, threat models, datasets, metrics, and experimental configurations.
- **Evaluation gap identification:** We identify critical methodological issues in defense evaluation, including the reliance on weak threat models, toy datasets, static triggers, and missing adaptive attacker evaluations.
- **Empirical evaluation:** We implement and evaluate 15 representative defenses across different categories (pre-training, in-training, post-training) against five diverse attacks (BadNets, Blended, WaNet, AdvDoor, Narcissus) on datasets of varying difficulty. Our results demonstrate that defense performance does not generalize from toy to realistic benchmarks. Our code is available at https://anonymous.4open.science/r/BackdoorDefensesAnalysis-A058/.
- **Actionable recommendations:** We provide concrete guidelines for defense researchers and practitioners, including minimum evaluation requirements (dataset difficulty, attack diversity, adaptive attacker consideration) and best practices for reproducible, fair comparison.
- **Public dataset:** Our comprehensive literature analysis is publicly available in our code repository as a dataset in a `.csv` file format, which is available for contributions from the community.

Our literature and experimental analysis aim to provide an objective and constructive examination of the current state of research on backdoor defenses. Rather than a critique of individual works, our goal is strictly to provide a clear, data-driven perspective on prevalent methodological trends and potential gaps within the literature, thereby fostering a shared commitment to enhancing the rigor and advancement of defense mechanisms. Moreover, the successful integration and reproduction of the representative defense methods selected for our experimental analysis speaks highly of their academic rigor and transparency.

12.2. BACKGROUND

12.2.1. BACKDOOR ATTACKS

Backdoor attacks are training-time attacks that add hidden behavior to the model [36]. The attacker's goal is two-fold: the first is to achieve high performance with clean (unaltered) samples, representing the clean behavior, and the second is to achieve high

Table 12.1: List of analyzed papers per year and conference. "/" denotes no papers, while N/A means the conference has not happened yet.

Year	Venue	Papers
2025	S&P	[139, 202, 129]
	NDSS	[122, 199, 190, 194, 102]
	ACM CCS	[43, 185, 70]
	USENIX Security	[114]
	NeurIPS	N/A
	ICML	[112, 95, 47, 105, 132, 119, 48]
	ICLR	[182, 1, 18, 55, 147, 204, 188, 192, 45]
	AAAI	[206, 39, 189, 74, 82]
	CVPR	[76, 173, 51, 198]
2024	S&P	[150, 65, 93, 99, 20]
	NDSS	[111, 120, 159, 27]
	ACM CCS	[187]
	USENIX Security	[75, 84, 135, 68]
	NeurIPS	[83, 94, 160, 179, 176, 97, 109, 59, 186, 113, 174, 98]
	ICML	[172, 91, 100, 183, 207, 33, 50, 73, 193]
	ICLR	[53, 178, 107, 171]
	AAAI	[208, 17, 117, 214, 153, 211, 2]
	CVPR	[37, 72]
2023	S&P	[10, 216, 158, 34, 69]
	NDSS	[21, 92, 19]
	ACM CCS	[170]
	USENIX Security	[108, 115, 29, 196, 152]
	NeurIPS	[58, 167, 164, 181, 154, 96, 161, 184, 215, 16, 133, 205, 142, 57, 63, 165, 131]
	ICML	[213, 168, 79, 66]
	ICLR	[64, 157, 41, 200]
	AAAI	[180, 23, 86, 191, 138]
	CVPR	[26, 137, 110, 101, 88, 144, 203]
2022	S&P	/
	NDSS	[121]
	ACM CCS	/
	USENIX Security	[125, 103]
	NeurIPS	[8, 13, 14, 156, 212, 87, 9, 151, 155, 209]
	ICML	[130]
	ICLR	[166, 40, 52, 56, 195]
	AAAI	[62]
	CVPR	[143, 90, 30, 38]
2021	S&P	/
	NDSS	[127, 11]
	ACM CCS	/
	USENIX Security	[141, 4]
	NeurIPS	[210, 163, 77, 136]
	ICML	[44, 169, 128]
	ICLR	[78, 71]
	AAAI	[106, 126]
	CVPR	
2020	S&P	/
	NDSS	/
	ACM CCS	/
	USENIX Security	/
	NeurIPS	/
	ICML	[123]
	ICLR	[25]
	AAAI	/
	CVPR	[67]
2019	S&P	[148]
	NDSS	/
	ACM CCS	[89]
	USENIX Security	/
	NeurIPS	[116]
	ICML	/
	ICLR	/
	AAAI	/
	CVPR	/
2018	S&P	/
	NDSS	/
	ACM CCS	/
	USENIX Security	/
	NeurIPS	[145]
	ICML	/
	ICLR	/
	AAAI	/
	CVPR	/

performance with malicious (modified) samples, representing the malicious behavior. Backdoor attacks can happen by poisoning the data [36], injecting malicious code (code poisoning) [5], or directly altering the model's weights (model poisoning) [49].

We briefly describe the relevant terminology of the backdoor attacks and their categorization; more information can be found in [80]. There are various methods for injecting a backdoor into a model. These can be broadly divided into dirty and clean label backdoor attacks. First, the attacker chooses a trigger and adds it to a subset of the training data. Formally, let the training dataset be denoted as $\mathscr{D} = \{(\mathbf{x}_i, y_i)\}_{i=0}^{N}$, where $\mathbf{x}_i \in \mathscr{X}$ represents an input sample and $y_i \in \mathscr{Y}$ its corresponding ground truth label. The attacker's chosen trigger t is added to a subset of training data. The proportion of poisoned data is given by $\epsilon = \frac{M}{N}$ where M is the number of altered samples and $M \ll N$. By adding the trigger on each chosen sample, the attacker creates a malicious sample $\hat{\mathbf{x}} = \varphi(\mathbf{x}, t)$, where φ is a trigger creation function. If the attacker follows a dirty label strategy, the ground truth label is changed to an attacker-chosen one $\hat{y}$ [36]. For a clean label strategy, the ground truth label remains unchanged [146]. The malicious data set is constructed as $\hat{\mathscr{D}} = \{(\hat{\mathbf{x}}_i, \hat{y}_i)\}_{i=0}^{M}$ for dirty label attacks or $\hat{\mathscr{D}} = \{(\hat{\mathbf{x}}_i, y_i)\}_{i=0}^{M}$ for clean label attacks.

Depending on the trigger generation function $\varphi(\cdot)$, triggers are classified as static or dynamic. The static triggers, also known as sample agnostic, remain the same regardless of the input sample. Dynamic triggers, also known as sample-specific, vary depending on the input, i.e., they are unique per sample. Depending on how we choose which samples to poison, we divide the attacks into source-agnostic and source-specific [42]. In source-agnostic attacks, the trigger activates the backdoor only when applied to a sample of a specific class. On the other hand, in the source-agnostic attacks, the trigger can activate the backdoor in any sample.

The attack's performance is measured through attack success rate (ASR), which is the number of backdoor activations over the number of triggered inputs fed to a poisoned model. Clean accuracy, on the other hand, represents the behavior of the model on clean samples.

12.2.2. BACKDOOR DEFENSES

Deployment Stage In the literature, backdoor defenses are categorized according to the stage of the machine learning pipeline at which they are applied [162]. This classification distinguishes defenses based on the defender's capabilities and the available resources. Specifically, we consider three main categories: *pre-training*, *in-training*, and *post-training* defenses [162]. *Pre-Training* defenses are applied before the model is trained. These defenses aim to prevent the injection of backdoors by sanitizing or augmenting the training data, selecting robust model architectures, or applying secure initialization techniques. The defender controls the dataset and training setup but has not yet started the actual training process. *In-Training* defenses control the training phase of the model. That is, the defender has access to the training data, the model, the weights, and the training hyperparameters. Furthermore, the defender controls the entire training procedure and can modify it to be robust against backdoors. *Post-Training* defenses analyze the model after it has been trained. These defenses assume that the defender does not participate in the training process but can analyze and modify the trained model.

Domain Defenses can be input-, feature-, and parameter-based, depending on where they are applied. *Input-space* defenses focus on the samples given to the model. Such defenses evaluate or check the model's input before passing it to the model. *Feature-space* defenses analyze the latent-space (at some point in the model), after the samples are passed to the model, to identify anomalies. *Parameter-space* defenses analyze the parameters of the model during training or after the training, and look for outliers or other indications that can be connected with a backdoor.

Execution Stage The defenses can also be categorized into two categories based on their deployment stage: *Online* and *Offline* [31]. *Online* defenses work during the execution time and detect malicious samples or behaviors, on the fly. These types of defense are often bound by execution time. *Offline* defenses, on the contrary, are countermeasures that do not have a time constraint and are executed in a detached mode from the main execution process, for instance, checking the dataset before training, or analyzing the model's weights before deployment.

Task Backdoor defenses may perform several actions to prevent backdoor attacks. For example, one approach might involve identifying malicious instances within the training data, whereas another could determine whether or not a model contains a backdoor. *Malicious Sample Detection* defenses inspect the training dataset and look for malicious samples. The defenses could then further clean the dataset by removing the trigger from the malicious sample or just flagging that sample as malicious and removing it. *Backdoor Detection* defenses aim to detect the backdoor during training or after the training is completed. *Backdoor Removal* defenses remove the backdoor from the trained model after the backdoor is detected.

12.2.3. Common Threat Models

Backdoor attacks occur during the training phase, creating diverse threat scenarios depending on the attacker and defender capabilities. We categorize threat models along a spectrum of access and control, from full system access (white-box) to minimal interaction (black-box).[1]

White-Box Both the defender and the attacker have full access to: (i) model architecture and weights; (ii) training algorithm, process, and hyperparameters; and (iii) training data (the defender has no knowledge of whether the data is poisoned). For defenders, this scenario may occur when they use publicly available datasets from untrusted sources (e.g., Kaggle) to train their models from scratch or fine-tune pre-trained models from community repositories (e.g., Hugging Face). For attackers, this scenario is possible when, using their own data, they inject a backdoor in their own model and then release it for public use.

Black-Box This represents the most restrictive threat scenario, in which both the defender and the adversary have severely limited access and control. The defender can only

[1] We provide some examples that do not represent the entire landscape of possible threat configurations.

interact with the model by providing inputs and observing the corresponding outputs, without any visibility into or ability to modify the model's internal parameters or architecture. Similarly, the adversary can influence the model only indirectly. For example, by submitting malicious inputs through public APIs or by manipulating physical objects in the environment (e.g., placing stickers on stop signs) with the goal of having these altered instances captured and included in the training data.

Gray-Box Anything that is not *white-box* or *black-box* belongs in this scenario. It contains a wide range of partial knowledge and capability combinations. For example, the defender has access to the model weights but cannot access the training data. Alternatively, an attacker can inject malicious samples into a subset of the training data, but cannot control the model or training procedure.

Across all the threat models, we assume that the defender has access to a clean holdout dataset, and a trusted set of benign samples not used during training, which will be used to measure the clean accuracy and as a reference for detecting anomalies. This is a standard assumption in the literature. In this work, we evaluate various defenses each with its own assumptions. As a result, we do not follow a specific threat model in our experiments.

12.3. LITERATURE STUDY

The research on backdoor defenses has gained significant attention in recent years. Between 2018[2] and 2025, the number of backdoor defense papers presented at AI and security conferences increased substantially. Very few defenses were proposed before 2020, but the numbers began to rise notably from 2021 onward. Although 2025 data are incomplete, current results already show continued interest.

Clearly, the number of published papers[3] motivates a more systematic evaluation of approaches given in the literature. Many novel and interesting works have been published in the last few years; however, there is no framework to help researchers evaluate the different aspects of each defense and perform a thorough robustness evaluation, resulting in incomplete reported results. In this work, our aim is to establish methodologies and best practices to examine backdoor defenses, helping researchers design better solutions and readers select appropriate defenses for each use case.

12.3.1. METHODOLOGY

We conducted an extensive literature review by evaluating all the papers on backdoor defenses published in selected A^* venues[4] across both the security (i.e., S&P, USENIX Security, NDSS, and CCS) and AI communities (i.e., ICLR, ICML, NeurIPS, CVPR, and AAAI) from 2018 to 2025. A total of 183 papers were manually analyzed, with key information extracted systematically from each study and saved in a `.csv` file.[5] The extracted

[2] We started our analysis from 2018, as the first backdoor attacks were BadNets [36] and Blended [15] both introduced in 2017, and the first countermeasures on this topic, i.e., Spectral Signatures [145] and Fine Pruning [85] were published in 2018.

[3] As we consider only nine conferences, the total number of backdoor defense papers is significantly higher.

[4] According to popular conference ranking systems, such as www.conferenceranks.com.

[5] We shared this file in the repository to allow further analysis and contributions from the community.

Table 12.2: Number of analyzed papers per year and conference. "-" denotes no papers, and N/A indicates that the conference has not yet occurred. The table contains 181 papers. However, in our analysis, we included STRIP [32] and Fine Pruning [85] since we observed that, while they are not published in any of the considered conferences, they are the second and the third most popular baseline defenses.

Year	S&P	NDSS	CCS	USENIX	NeurIPS	ICML	ICLR	AAAI	CVPR
2025	3	5	2	1	N/A	7	9	5	4
2024	5	4	1	4	12	9	4	7	2
2023	5	3	1	5	17	4	4	5	7
2022	-	1	-	2	10	1	5	1	4
2021	-	2	-	2	4	3	2	2	-
2020	-	-	-	-	-	1	1	-	1
2019	1	-	1	-	1	-	-	-	-
2018	-	-	-	-	1	-	-	-	-

information contains:

- **Identification information**: defense name, conference name, and publication year.
- **Evaluation information**: defense's code availability, the model architectures and the datasets (along with their domains, e.g., images, text, graph, sound, etc.) used in the experiments, the baseline defenses that were used for comparison, and the types of tested backdoor attacks. To gain a detailed view of the attacks against which the proposed defense is evaluated, we keep track of the different attack characteristics, whether an adaptive attacker was tested, the names of the attacks, the assumptions related to the attacker's knowledge of the defense, and the poisoning rates used.
- **Evaluation metrics**: the metrics reported by the authors in their experiments, organized in two categories: attack performance with or without defense, and the attack's stealthiness. Finally, we report whether clean and no-attack performance were evaluated.
- **Defense properties**: information needed to characterize a defense as discussed in Section 12.2.2. More specifically, the defense's category, domain, and deployment stage.
- **Defender's knowledge**: the scenario where the defense is evaluated (e.g., Federated Learning), the defender's knowledge about the model (architecture, hyperparameters, and weights), the backdoor's existence, the trigger (e.g., BadNets), the trigger's properties (e.g., trigger size or location), the trigger's distribution, and if the attack is targeted/untargeted.
- **Access to resources**: whether the defender has access to poisoned or clean data, the model's weights, or the feature maps. Additionally, we check whether the defense requires additional labeled or unlabeled data beyond the training set, or extra models (poisoned or clean) beyond the one evaluated.
- **Defender's capability**: whether the defender can train surrogate models, alter the model's inputs or weights, fine-tune the model, and filter or purify malicious data.

Observe that the last three categories are related to the defender's threat model.

12.3.2. ANALYSIS OF THE COLLECTED DATA

We present now relevant statistics from the collected information about existing defenses to support our discussion and experimental choices for this work, as well as to highlight current trends. We organize our findings into two categories: Recommendations (R), presented in light gray boxes, which are actionable suggestions with known solutions calling for widespread implementation and adoption; and Challenges (C), appearing in darker gray boxes, which are observed shortcomings requiring further research.

We note that ≈ 80% of the considered publications provide source code. Although this percentage is high, this still highlights the fact that many papers do not release their source code, while it greatly simplifies reproduction and validation, and enables extensive experimental analysis like the one we report in Section 12.5.

R1: Every published backdoor defense (and attack) paper should provide source code.

Focusing on the deployment phase, we found that most defenses fall into the post-training category, accounting for nearly 60% of all analyzed approaches. In-training defenses represent the second-largest group, accounting for 33% of the total. In contrast, pre-training defenses and hybrid methods, such as those combining pre- or in-training with post-training techniques, are relatively rare, less than 5%. Finally, less than 1% considers all stages of the training process. These different categories assume different attack and defense threat models and capabilities. Post-training is commonly considered a worst-case scenario for defenders as it is a reactive stage (end-user) where defense assumes the attack has already succeeded. Thus, the trend of focusing on a reactive defender neglects more effective stages for defending against backdoor attacks (pre- and in-training).

R2: Balance the study of different defense deployment phases by focusing on proactive defense methods.

Moreover, the diverse settings of existing backdoor attacks require defenses that are flexible in different deployment scenarios, and limiting a defense to operate in a single phase in most cases requires assumptions that may be unrealistic in certain scenarios. Nowadays, we rely heavily on pre-trained models for transfer learning, which may have been trained on malicious data inaccessible to the user downloading them, making pre-training defenses impractical. At the same time, when we have access only to data and must rely on online services for "black-box" training or fine-tuning, due to limited resources or the use of closed models (e.g., OpenAI[6]), we cannot control or inspect the resulting model and can only use it as-is. This emphasizes the need for defenses that remain effective and adaptable across different phases.

If we consider defenses grouped by categories as shown in Figure 12.1, we can see how defenses in each category use different resources. A common requirement is having auxiliary models, which could not be realistic due to computational constraints when dealing with large models or datasets, since they have to be trained, often from scratch.

[6] https://openai.com/index/gpt-4o-fine-tuning/

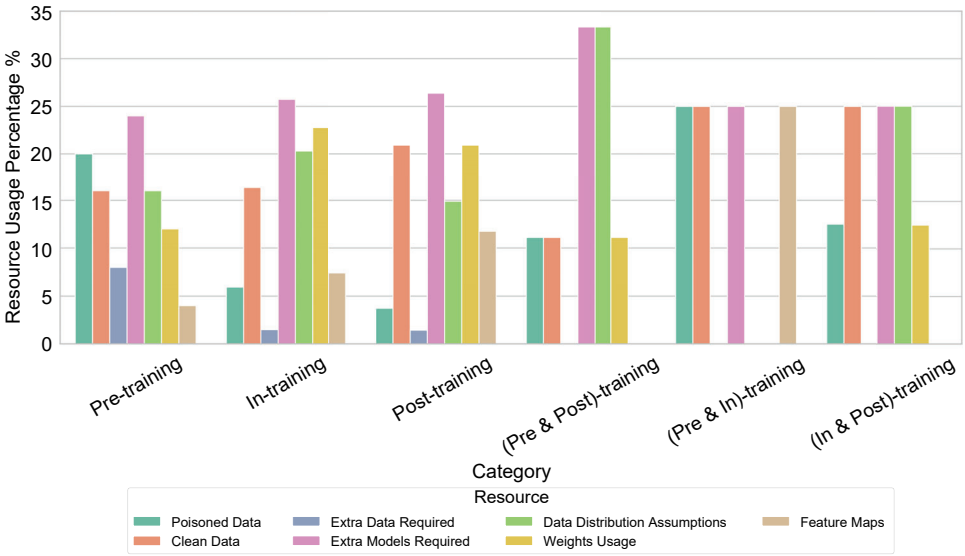


Figure 12.1: Distribution of resources across different training phases, showing the relative occurrence of each resource type during pre-, in-, and post-training phases.

This aspect is further discussed in Section 12.5 where we show how this could limit the practical use of these defenses. The second most used resource for pre-training defenses is the reliance on poisoned data. This constitutes a significant assumption in practical scenarios because it requires access to every potential attack to ensure resilience, thereby restricting the generalizability of defenses.

Finally, many methods assume access to clean training data, an unrealistic assumption in real-world settings where models are distributed to large user communities, as seen with the HuggingFace repositories. In such cases, a more realistic assumption would imply access to data exhibiting the same feature distribution as the training set, without requiring the exact data points used during training.

C1: Develop scalable defenses independent of auxiliary models and poisoned/-clean training data, using alternative resources (e.g., feature maps or same-distribution samples not from the training set) to improve generalization and real-world performance.

In Figure 12.2, we provide statistics on the number of datasets, attacks, types of models, and baseline defenses used to evaluate performance. As shown in Figure 12.2a, most defenses are tested using three datasets, while considering more than six datasets is rare. Similarly, in most cases, one to three different model types are used (see Figure 12.2d). In Table 12.3, we provide examples of the grouping we used to separate different model types for analysis. The performance comparison is done against five or six baseline defenses,[7]

[7]Notably, in our sample, some papers do not even compare against any defense [11, 154, 25, 23]. In one case

Table 12.3: Classification of neural network models into architecture families for systematic analysis.

NN Group Type	Example Variants to Include
ResNet	ResNet, ResNet-18, ResNet-34, ResNet-50, ResNet-101, WideResNet, PreActResNet
VGG	VGG, VGG-11, VGG-16, VGG-19
ViT	ViT, DeiT, CLIP-ViT, Swin Transformer
CNN	CNN, ConvNet, Custom CNN
GCN	GCN, GAT, GraphSAGE, GIN, RGCN, GatedGCN
MobileNet	MobileNet, MobileNetV2
Transformer	BERT, RoBERTa, GPT, LLaMA, ALBERT, DistilBERT, GPT-2, GPT-3, GPT-4, T5
DenseNet	DenseNet, DenseNet-121, DenseNet-169, DenseNet-161
LeNet	Inception, GoogLeNet, LeNet, LeNet-5
AlexNet	AlexNet
LSTM/GRU	LSTM, GRU, Bi-LSTM
MLP	MLP, FC-NN
Diffusion	DDPM, EDM, Stable Diffusion, DDIM
Reinforcement Learning	PPO, DRL, Policy Learning NN
Detection Models	Faster R-CNN, YOLO, SSD, DETR

and the number of targeted attacks is typically between one and five.

Finally, we observed that, overall, only about 58% of the proposed defenses consider an adaptive attacker, around 55% of the defenses presented at AI-focused conferences, and roughly 67% of those presented at security conferences. While considering this aspect is crucial, given that adaptive attacks are generally the most advanced, it alone does not suffice to demonstrate the effectiveness of the defense. Overall, this highlights the heterogeneity of evaluation settings, which limits the comparability of results across studies and underscores the need for more comprehensive and standardized benchmarking.

Another important trend visible in Figure 12.3a is the strong tendency to consider a narrow subset of available benchmark datasets. In particular, CIFAR-10 emerges as the predominant choice, serving as the baseline for most evaluations. While this facilitates comparability across studies, it also raises important concerns about the general applicability of the evaluation results for the proposed defenses.

Figure 12.3d shows the most common attacks used in the experimental evaluations of the proposed defenses. Among these, Badnets and Blend attacks are the most common ones. The trend highlights a continued reliance on a narrow and well-established (though arguably outdated) set of evaluation techniques. Despite the rapid growth of the literature on adversarial machine learning, only a small fraction of defenses evaluate their approaches on recent attacks. Among the five most frequently considered attacks, the most recent is SIG [28], introduced in 2021, thus highlighting a gap between the development of new attacks and their adoption in defense evaluations. Therefore, while defenses may appear robust under current experimental standards, their true resilience against state-of-the-art or emerging attacks remains underexplored. Most attacks discussed in Section 12.3.1 are dirty-label, with clean-label attacks largely overlooked. Additionally, triggers are typically visible rather than stealthy, and only one case involved a dynamically generated trigger, highlighting limited attention to complex or adaptive designs. The same

also, the proposed theoretical framework against data poisoning was not tested against any specific attack, but only against distribution shift [154].

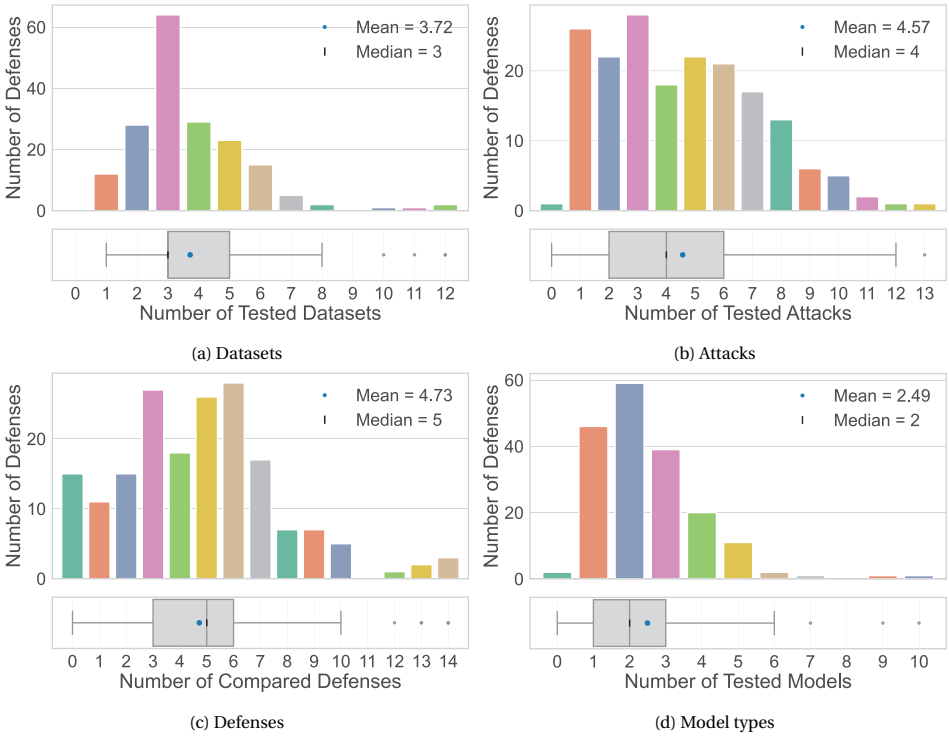
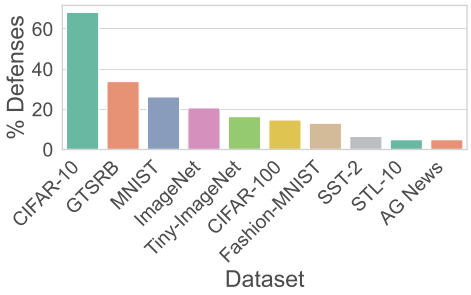


Figure 12.2: Distribution of the number of used datasets, attacks, defenses, and model types used in the analyzed defenses.

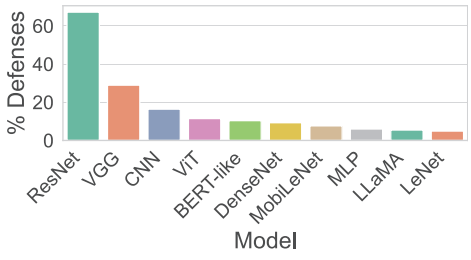
observation also holds for the baseline defenses most frequently used for comparison with the proposed methods. Notably, the most recent of these is ABL [77] from 2021, also in this case.

In Figure 12.3b, we examine the machine learning models used for the evaluation of the analyzed defenses.[8] Typically, used models belong to the ResNet group that includes different variants of ResNet models, such as ResNet18 or ResNet50. This raises the concern that defenses may be overly tuned to a specific architecture, limiting their adaptability across different model families. Because of this, defenses might potentially exploit unique characteristics of specific target models, such as residual blocks or simple CNNs layers, while neglecting a wider set of characteristics from diverse model architectures. As a consequence, this can effectively impose an expiration date on the defense's applicability. Additionally, we observe a preference for simpler models, which may require less time and resources to train. From Figure 12.3, we can conclude that focusing only on a limited selection of toy datasets, specific model families, or certain attacks and defenses can lead to defenses that are overfitted to these particular scenarios, limiting their applicability beyond these predefined assumptions.

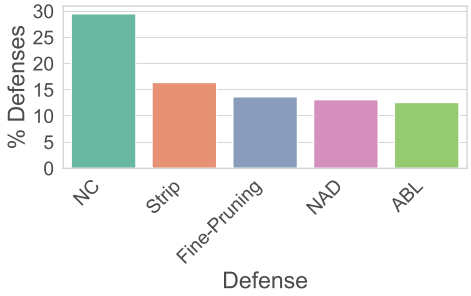
[8] See Table 12.3 for the model groups.



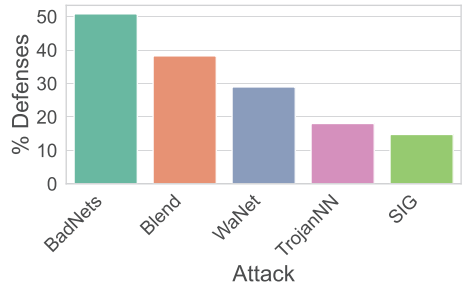
(a) Top 10 most used datasets.



(b) Top 10 most used models.



(c) Top 5 most evaluated defenses.



(d) Top 5 most tested attacks.

Figure 12.3: Overview of dataset, attack, defense, and model evaluation statistics.

C2: Design defenses that operate at a higher level of abstraction, avoiding assumptions tied to specific architectures, such as CNNs, or data, to ensure future-proof robustness across models and domains.

Thus, the number of evaluated attacks/defenses, models, and datasets should be less important than ensuring that diverse categories are covered. This implies using both simple datasets (e.g., MNIST) and complex ones (e.g., ImageNet-1K), as well as simpler model architectures (e.g., VGG family) and more advanced ones (e.g., ViT), as proved by the results we collected in Section 12.5. Finally, for attacks and defenses, it is important to use both well-established baselines and recent state-of-the-art results.

R3: Prioritize diversity in the evaluation setup by including datasets of varying complexity, models from different families, and both established and recent attacks and defenses, rather than simply increasing their number.

Next, we examine the distribution of defenses based on the domain of the datasets considered in the paper. We observed that the image domain is the most frequently used, with the text domain receiving more attention starting from 2022. Other domains are not as prominent, although in recent years, we can observe a greater diversity in the dataset domain than in earlier years. This variation could be attributed to the rise of Large

Language Models (LLMs), though the prevalence of the image domain still remains. As LLMs continue to advance and evolve into multi-modal systems such as visual language models (VLMs), which integrate both textual and visual inputs, the boundaries between input domains are becoming increasingly blurred, making domain-specific defenses less appealing.

C3: Design defenses that are dataset domain-agnostic.

Table 12.4: Percentage of defenses per defense domain that can work over a Single or Multiple dataset domains (absolute count in parentheses).

Defense Domain	Single (count: 140)	Multiple (count: 20)
Input-Space	48.6%	20.0%
Feature-Space	23.6%	15.0%
Parameter-Space	44.3%	70.0%

We then analyze the relationship between the action space of the defenses (Input, Feature, or Parameter) and the dataset domain (e.g., images or text) in Table 12.4. Defenses that operate on model parameters are more frequently tested against a wider range of dataset domains, such as both image and text. This suggests that access to model parameters allows for the development of defenses that are more broadly applicable across different types of data. While designing defenses that operate across domains is valuable, overly general approaches can reduce effectiveness. In many industrial settings, more specialized defenses, tailored to a specific application, often achieve higher reliability. Therefore, a careful trade-off between generality and specificity should be considered when developing new defenses.

R4: Combine general, model-agnostic defenses with domain and pipeline-specific methods to achieve transferability.

Figure 12.4 depicts the usage patterns of evaluation metrics across defenses. We can broadly categorize metrics into three groups: **(i) task performance:** Clean Accuracy (CA), ASR, and Certified Accuracy; **(ii) detection capability:** Detection Accuracy, Precision, Recall (TPR), False Positive Rate (FPR), F1-Score, and AUC-ROC; **(iii) efficiency:** Execution Time. From our analysis, we found that most adopted metrics are ASR, followed by CA in AI conferences, and a set of Other different metrics and CA at security conferences. The difference is quite large, with $\approx 65\%$ and $\approx 80\%$ of defenses reporting ASR, but only $\approx 48\%$ and $\approx 39\%$ report CA at AI and Security conferences, respectively. The absence of a standardized and consolidated set of evaluation metrics limits the comprehensive comparison of proposed defenses. Although ASR serves as an indicator of a defense's capacity to block attacks, its true effectiveness also depends on the ability to maintain the model's main task performance (commonly measured with CA) and ensure seamless deployment. Moreover, defense evaluation papers rarely report execution time, despite its critical importance for practical deployment.

Stealthiness is a crucial property of a backdoor attack, tightly connected to the attack's

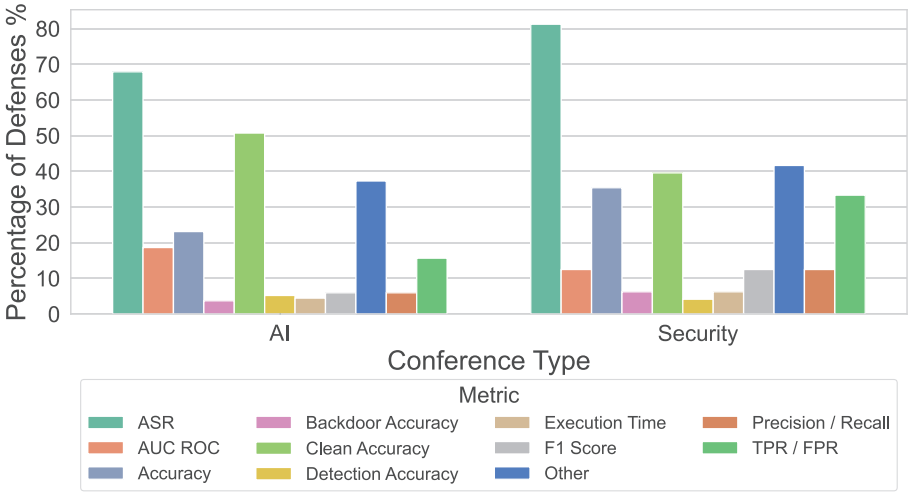


Figure 12.4: Usage patterns of evaluation metrics across defenses by conference type. "Other" includes metrics that were rarely reported and fall outside the ten most frequently used ones.

performance against a defense. An attack could be stealthy in the input-, feature-, or parameter-space, and many defenses are not effective in all spaces simultaneously [177]. Thus, to verify a defense's generalization, it is important to consider attacks that are stealthy in different spaces. The evaluator could identify stealthy attacks using various stealthiness metrics or pick attacks that are known to be stealthy in different spaces. For example, Adap-blend is stealthy in the feature space but not in the input space, WaNet is stealthy in the input space but not in the feature space, and Grond is stealthy in all spaces [177]. Surprisingly, only 9 out of 183 surveyed papers report results about the attack's stealthiness, which shows that this topic is often overlooked in a defense evaluation.

R5: When evaluating a defense, we need to consider attacks with different stealthiness capabilities (pixel space, latent space, parameter space, frequency domain), often measured by appropriate metrics, instead of testing many attacks with the same properties.

Defenses must report execution time, a critical metric for practical deployment. Some theoretically effective defenses may fail in real-world applications due to high computational costs, yet this limitation remains unclear without systematic runtime reporting and comparison. Following Recommendation R6 enables fair comparison of computational costs and helps practitioners assess deployment feasibility.

R6: Report the defense's execution time and compare it against the baseline (no-defense) execution time, quantifying the induced overhead under identical hardware settings.

Depending on the nature of the defense, e.g., backdoor detection (model or input-level) or backdoor removal, different metrics will be reported as applicable. However, while we observe some commonly used metrics, there are still a number of papers reporting "other" metrics, which include less used ones, such as "anomaly index". These can be valid, as defense might be further from the already established types of defenses, but we might want to avoid arbitrary metrics. While core metrics (accuracy and ASR) are widely adopted, inconsistent naming conventions and the introduction of non-standard metrics without clear justification complicate reproducibility and cross-study comparisons.

C4: Lack of standardized terminology and consistent reporting of metrics when evaluating backdoor defenses.

Next, Figure 12.5 provides statistics (based on what the authors reported) on what happens once the defense is deployed on clean models. We can observe that in most cases, the effect is not discussed in the paper. Over the years, more papers have started to consider this, reporting (vague) information on it.

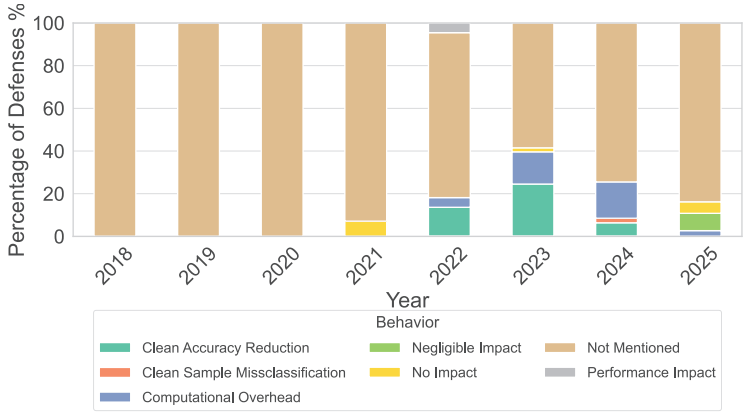


Figure 12.5: Percentage per year of defenses that consider the effect of the defense when there is no attack.

R7: Report the impact of the defense without any backdoor present, explicitly including the main task (clean) accuracy when the defense is applied.

12.4. Experimental Setup

Datasets We evaluate on MNIST, CIFAR-100, and ImageNet-1K, selected for their diversity and difficulty (as in Recommendation R3). MNIST is grayscale and simple; its inclusion ensures that grayscale images are considered in the evaluation, as such data frequently occurs in applications like medical imaging. CIFAR-100 offers mid-level complexity with 100 classes. ImageNet-1K provides a large-scale, realistic benchmark with 1M+ images across 1 000 classes.

Models As seen in Figure 12.3, the most used models are ResNet, VGG, and the ViT family, followed by custom CNN architectures and DenseNet. Thus, in our experiments, we consider ResNet-18 [46], VGG-19 [134], ViT-B/16 [24], and DenseNet-121 [54]. These cover both convolutional and transformer-based architectures.

Defenses Defenses are selected based on **(i) code availability:** we include only defenses with publicly available, runnable implementations to ensure reproducible evaluation and avoid errors; **(ii) category coverage:** we select at least one defense from each major category combination (see Section 12.2.2) to provide comprehensive coverage of defense strategies; **(iii) recency:** when multiple defenses occupy the same category, we favor more recent work (2022-2024) to reflect the current state-of-the-art, unless older seminal work represents a fundamentally different approach; **(iv) integration complexity:** we prioritize defenses with standard interfaces and minimal implementation dependencies to enable fair comparisons. See Table 12.5 for a complete list of the defenses considered for our experimentation.

Table 12.5: Selected defenses categorized by training stage, inspection domain, and execution mode.

Deployment Stage	Inspection Domain	Execution Mode	Defense
In-training	Feature	Offline	Spectral Signatures [145]
	Feature	Online	Trap and Replace [151]
	Input	Online	Reback (Need for Speed) [93]
	Input, Parameter	Online	PDB [160]
	Parameter	Offline	ABL [77]
	Parameter	Online	CBD [203]
Post-training	Feature	Offline	NAD [78]
	Feature, Input	Offline	Unicorn [157]
	Input	Offline	Neural Cleanse [148]
	Input	Online	Scale-Up [41]
	Input, Parameter	Offline	AIBD [189]
	Parameter	Offline	IBAU, BAN [175]
	Parameter	Online	BadExpert [171]
Pre-training	Input	Offline	Incompatibility Clustering [64]
	Input, Parameter	Offline	Metasift [196]

Backdoor Attacks We evaluate defenses against five representative attacks (Table 12.6): BadNets [36], Blended [15], WaNet [104], AdvDoor [201], and Narcissus [197], covering both input- and feature-space manipulations, and dirty- vs. clean-label settings.

Poison rates We selected four specific poison rates, 0.005 (0.5%), 0.01 (1%), 0.05 (5%), and 0.2 (20%), to comprehensively evaluate selected defenses. The rates of 1% and 5% were included because our analysis shows they are the most commonly used poisoning

Table 12.6: Summary of utilized backdoor attacks.

Attack	Venue	Year	Type	Trigger Type
BadNets [36]	IEEE Access	2017	Dirty	Patch-based (static)
Blended [15]	arXiv	2017	Dirty	Image blending (transparent overlay)
WaNet [104]	IEEE S&P	2021	Dirty	Spatial warping (imperceptible)
AdvDoor [201]	AAAI	2022	Dirty	Universal adversarial perturbation
Narcissus [197]	CVPR	2023	Clean	Sample-specific

Table 12.7: Average CA/ASR(%) without defense.

Dataset	Model	AdvDoor	BadNets	Blended	Narcissus	WaNet	No Attack
MNIST	DenseNet-121	99.21 / 4.65	99.18 / 99.99	99.25 / 100.0	99.34 / 0.08	76.78 / 99.92	99.14 / N/A
	ResNet-18	98.94 / 5.85	99.07 / 99.91	99.03 / 100.0	99.05 / 0.07	98.84 / 56.41	98.9 / N/A
	VGG-19	99.01 / 2.74	99.15 / 99.98	99.09 / 99.99	99.1 / 0.13	97.76 / 99.33	98.96 / N/A
	ViT-B/16	97.96 / 2.84	98.02 / 99.96	98.08 / 99.96	98.1 / 0.17	97.72 / 54.36	98.06 / N/A
CIFAR-100	DenseNet-121	78.51 / 99.96	70.39 / 99.33	69.43 / 49.62	78.97 / 82.39	68.96 / 45.29	79.09 / N/A
	ResNet-18	76.1 / 99.22	68.52 / 97.41	66.41 / 51.24	76.56 / 83.96	67.05 / 47.87	76.87 / N/A
	VGG-19	71.44 / 99.73	71.8 / 98.04	54.42 / 37.02	72.05 / 61.29	51.65 / 34.15	72.44 / N/A
	ViT-B/16	91.12 / 95.76	91.2 / 98.93	91.19 / 99.46	91.05 / 47.19	90.56 / 45.5	91.09 / N/A
ImageNet1K	DenseNet-121	69.65 / 0.02	70.48 / 79.82	69.41 / 89.79	69.52 / 0.99	69.07 / 68.14	69.57 / N/A
	ResNet-18	64.21 / 0.02	65.15 / 79.66	64.08 / 84.67	64.25 / 0.72	63.77 / 49.01	57.72 / N/A
	VGG-19	67.87 / 0.07	68.44 / 83.28	67.48 / 87.16	67.54 / 1.72	67.62 / 49.11	67.7 / N/A
	ViT-B/16	82.34 / 0.0	81.01 / 79.85	82.27 / 90.78	82.44 / 0.22	82.12 / 47.91	82.41 / N/A

levels in the existing literature. We then add two edge-case rates: 0.5%, representing a subtle attack used in a minority of defense papers (5%), and 20%, representing a less subtle attack (used in 10% of defense papers). We evaluated Narcissus with three poison rates (other attacks used five), because Narcissus is a clean-label attack. Matching the total number of poisoned images requires increasing the per-class poison rate, which is limited by class size (e.g., poisoning an entire class in CIFAR-100 equals only 1% of the dataset). In the case of Imagenet-1K, we could not achieve a poison rate matching the baselines, even when poisoning an entire class, due to the large number of classes. We also test poison rate 0, indicating a no-attack scenario, to evaluate the impact of the deployed defenses when the attack is not present.

12.5. Experimental Results

Excluding runs with timeouts and Unicorn and Neural Cleanse defenses, the 2 798 successful experiments across all three datasets collectively required more than 170 days of total compute time. All results are available in the repository[9], while here we report the key results. The missing values in the presented results mean that the given defense failed in that specific configuration. In Table 12.7, we report the mean CA and ASR for the considered backdoor attacks and for clean training without an attack. These provide baselines for fair comparison and analysis in this Section.

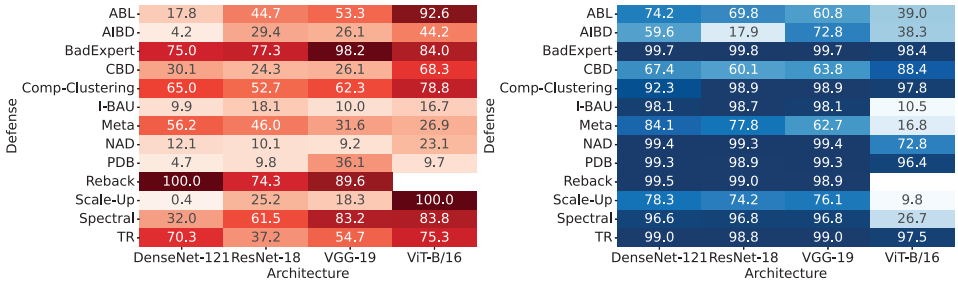
[9] https://anonymous.4open.science/r/BackdoorDefensesAnalysis-A058/.

Model Architectures Based on the survey in Section 12.3, most existing defenses are developed for image-based attacks on CNN architectures. Transformer-based models, such as Vision Transformers (ViTs), which underpin many state-of-the-art generative models, remain largely unexplored. Given their growing importance, ViTs should serve as key baselines for evaluating backdoor defenses, highlighting the need for architecture-agnostic defenses.

Observations. Examining the results of defenses grouped and averaged by architecture across MNIST, CIFAR-100, and ImageNet-1K (Figures 12.6 to 12.8) shows that the dominant pattern is that dataset complexity amplifies the CA–ASR trade-off: moving from MNIST to CIFAR-100 to ImageNet-1K, average CA drops sharply and many defenses fail on ImageNet-1K (several rows show CA ≈ 0). On MNIST and CIFAR-100, PDB and NAD consistently deliver the best joint outcomes (high CA with low–moderate ASR), with PDB edging MNIST (e.g., CA ≈ 99, ASR < 10) and NAD leading CIFAR-100 overall (CA in the mid-60s to mid-70s on CNNs with ASR ≈ 25). In contrast, methods like BadExpert, Comp-Clustering, and Spectral often preserve CA, but the defense is not effective as it does not decrease the ASR. On ImageNet-1K, only Metasift scales, as it achieves near-zero ASR (≤ 1) while keeping high CA (roughly 66–80) across all architectures. Everything else is either leaky (high ASR) or destructive (near-zero CA). We observe that for CIFAR-100, TR is excellent on DenseNet-121 (ASR ≈ 7 with CA ≈ 63), yet unstable elsewhere. Specifically, for ViT-B/16, the CA is kept relatively high, but the ASR drops only to ≈ 51, illustrating CNN-specific tuning. Moreover, this is visible in the MNIST results (Figure 12.6), as the CA is kept high for all CNN-based models, but is significantly negatively affected for many defenses with ViT-B/16. PDB and NAD are comparatively robust across CNNs (DenseNet/ResNet/VGG) on MNIST/CIFAR-100, while several methods (I-BAU, NAD, Reback) severely degrade CA on ImageNet regardless of model. Interestingly, ViT-B/16 is the weakest on CIFAR-100 for many defenses, but becomes the most usable on ImageNet when paired with Meta (high CA, ASR ≈ 0), suggesting capacity/inductive-bias help only if the defense scales. Overall, defense effectiveness is tightly coupled with both architecture and dataset scale: CNNs benefit from spatially grounded, input- or parameter-level defenses, whereas ViTs require strategies that exploit global patterns to preserve CA while mitigating attacks.

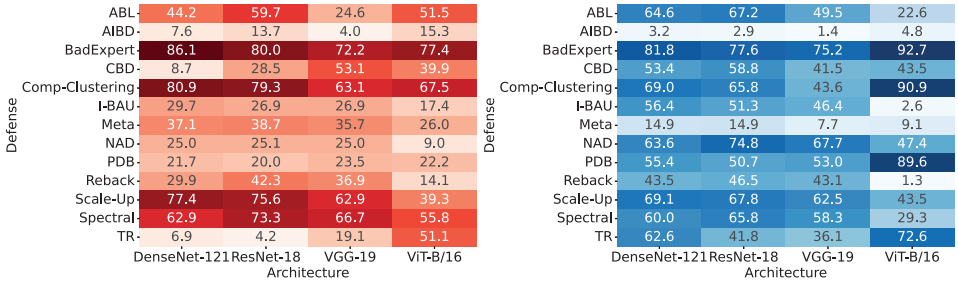
C5: While aiming to be model- and dataset-agnostic, current defenses exhibit strong sensitivity to both. Future research should aim to design truly architecture-agnostic defense frameworks that generalize across model families and datasets.

Trigger Types Based on the review in Section 12.3, most defenses are evaluated primarily against legacy, simple, and highly visible attacks. Only a few works consider advanced, stealthy triggers, such as adaptive or clean-label attacks, which are essential for realistic threat modeling. We also observe that some defenses underperform even in scenarios considered "too simple" to challenge them. The common assumption that a defense effective under worst-case conditions will also succeed in simpler ones does not always hold. For instance, in Figure 12.10a, the compatibility clustering fails against BadNets (ASR 93.8%).



(a) ASR (%) (b) CA (%)

Figure 12.6: Average CA and ASR for MNIST for different models.

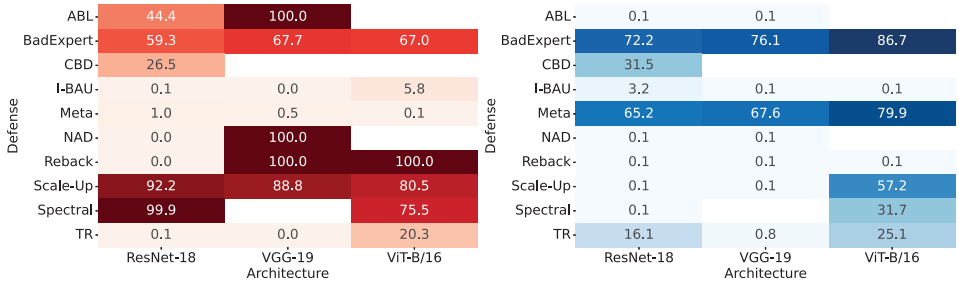


(a) ASR (%) (b) CA (%)

Figure 12.7: Average CA and ASR for CIFAR-100 for different models.

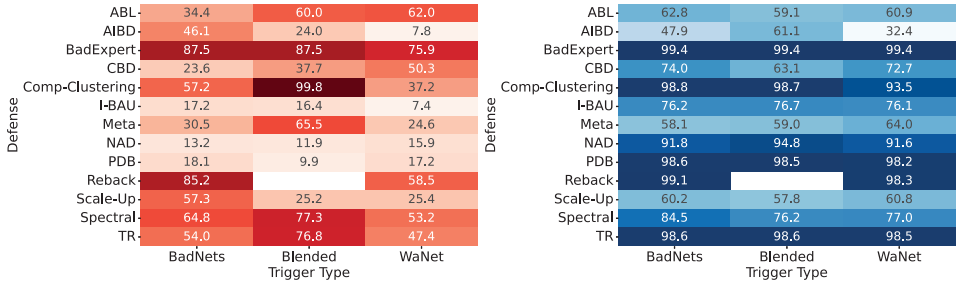
R8: Defenses need to be evaluated under various difficulty conditions, including simpler attacks, but also worst-case scenarios and adaptive attacker settings.

Observations. Analyzing defenses grouped and averaged by trigger types, Figures 12.9 to 12.11, reveal that on MNIST, the trigger choice (BadNets/Blended/WaNet) has little effect: NAD and PDB consistently achieve low ASR (≈ 10–18%) while keeping CA very high (≈ 92–99%), and I-BAU can push ASR lower on WaNet at the cost of CA. On CIFAR-100, trigger structure becomes decisive: localized/additive triggers remain tractable (PDB/NAD drive ASR to ≤ 1% with CA in the 60–70% range), while semantically integrated, clean-label trigger (Narcissus) evade pixel/parameter-level inspection. Methods that minimize ASR on these hard triggers (e.g., AIBD, Metasift) do so by crushing CA to single digits, while TR yields a somewhat usable trade-off (e.g., Narcissus ≈ 22% ASR at ≈ 53% CA). Outlier-filtering approaches (Spectral, Scale-Up) often preserve CA but also leave high ASR, meaning that the backdoors remain active on the non-filtered samples. On ImageNet-1K, defense scalability overwhelms trigger differences: Metasift uniquely sustains high CA (≈ 71%) with near-zero ASR (≤ 0.7%) across BadNets/Blended/WaNet, while alternatives either leak (e.g., BadExpert except on WaNet) or collapse CA (e.g., I-BAU, NAD, ABL). Note that missing entries indicate timeouts/failures, underscoring non-scalability. Overall, simpler, spatially bounded triggers are easier to detect, while



(a) ASR (%) (b) CA (%)

Figure 12.8: Average CA and ASR for ImageNet-1K for different models.



(a) ASR (%) (b) CA (%)

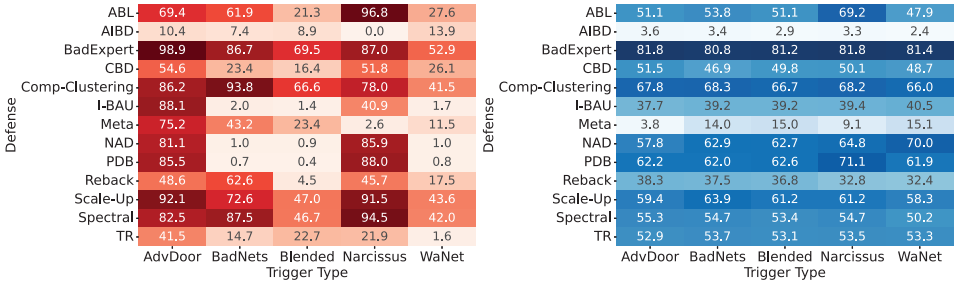
Figure 12.9: Average CA and ASR for MNIST for different trigger types.

semantically integrated or high-visibility triggers increasingly challenge defenses as model and dataset complexity grow.

C6: Existing defenses fail to generalize across spatial, semantic, or high-visibility patterns, especially on large-scale datasets.

Poisoning Rates Section 12.3 shows that evaluation practices lack standardization. Researchers often use arbitrary, typically low poisoning rates without a common benchmark, limiting cross-study comparability and biasing results toward defenses. Many also omit reporting attack success without defenses.

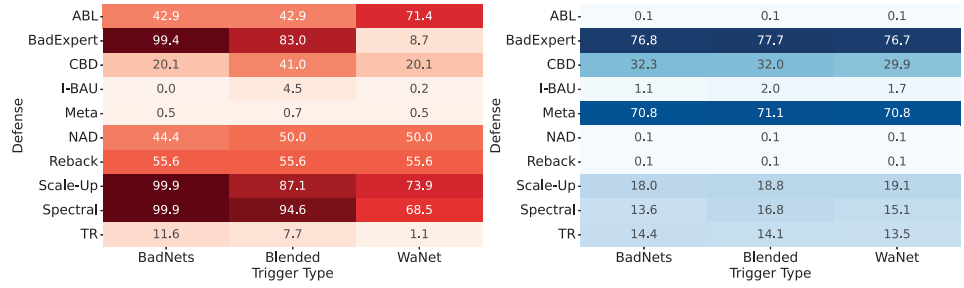
Observations. In Figures 12.12 to 12.14 we examine defenses grouped and averaged by increasing poisoning rates, highlighting the sensitivity of methods to both dataset complexity and attack strength. On MNIST, low poisoning rates (0.005-0.05) are often mitigated by parameter-level defenses (e.g., PDB, CBD, and IBAU), preserving CA while reducing the ASR. Higher rates (0.2) degrade CA and allow elevated ASR, revealing limits of defenses when attacks become pervasive. On CIFAR-100, even modest poisoning significantly reduces defense effectiveness: many methods see sharp ASR increases and CA drops, especially for complex attacks and deeper models. Large-scale datasets like ImageNet-1K worsen this effect, where high-resolution and semantically diverse inputs



(a) ASR (%)

(b) CA (%)

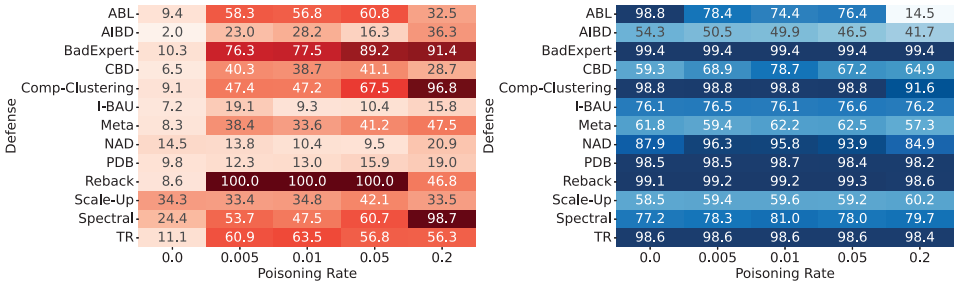
Figure 12.10: Average CA and ASR for CIFAR-100 for different trigger types.



(a) ASR (%)

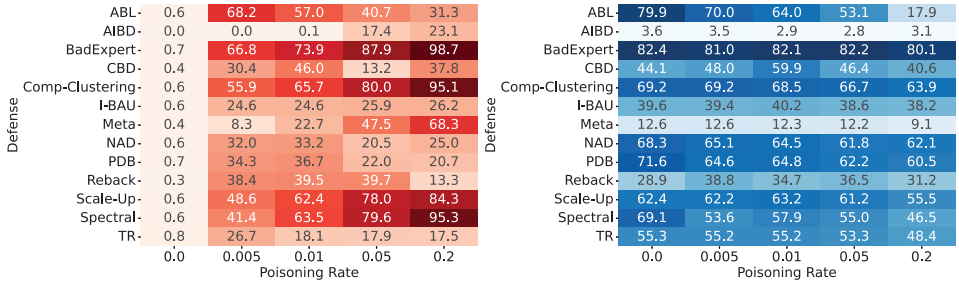
(b) CA (%)

Figure 12.11: Average CA and ASR for ImageNet-1K for different trigger types.



(a) ASR (%) (b) CA (%)

Figure 12.12: Average CA and ASR for MNIST for different poisoning rates.



(a) ASR (%) (b) CA (%)

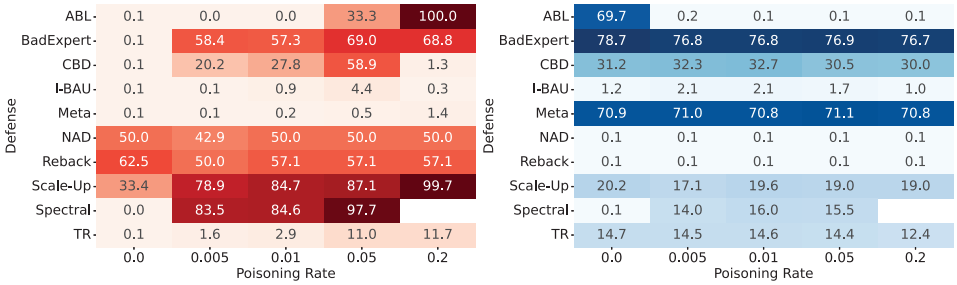
Figure 12.13: Average CA and ASR for CIFAR-100 for different poisoning rates.

make most defenses fragile against even low poisoning. It is interesting to see how, even in the absence of attacks, many defenses sensibly decrease the CA of the model, affecting its functionalities, giving further support for Recommendation R7. The initial concept highlights that defenses, although designed to enhance robustness, can inadvertently harm normal performance. Overall, robustness declines nonlinearly with the poisoning rate, with measurable impacts appearing even without attacks, and the interaction between model and dataset scale determines how effectively a defense preserves performance.

C7: Ensuring consistent robustness of the defense across poisoning strengths without sacrificing clean performance remains an open challenge.

Hyperparameter Selection Based on the review of the papers for this SoK and the re-implementation from the official repositories to perform experiments, the defenses are designed to work under different conditions, using different hyperparameters. Most of them were estimated experimentally without providing a clear strategy to choose them in an unknown scenario. Thus, in our experiments, we used the default hyperparameters defined in their official code.

Observations. To systematically quantify the adaptability of a defense across datasets, we introduce the Mean Absolute Difference Score (MADS). This metric measures how



(a) ASR (%) (b) CA (%)

Figure 12.14: Average CA and ASR for ImageNet-1K for different poisoning rates.

consistent the ASR of a defense remains when tested across multiple datasets under the same experimental settings. For a given defense d and experiment e, let $\mathrm{ASR}_{d,e}^{D_i}$ denote the ASR on dataset D_i, and let

$$\mu_{d,e} = \frac{1}{|D|} \sum_{i=1}^{|D|} \mathrm{ASR}_{d,e}^{D_i}$$

be the mean ASR across all datasets D. The MADS for defense d is then defined as the average mean absolute deviation across experiments:

$$\mathrm{MADS}_d = \frac{1}{EX \cdot |D|} \sum_{e=1}^{EX} \sum_{i=1}^{|D|} \left| \mathrm{ASR}_{d,e}^{D_i} - \mu_{d,e} \right|,$$

where EX is the number of experiments. A lower MADS indicates that a defense's ASR is more consistent across datasets, while a higher MADS indicates greater variability. In Figure 12.15, the results exhibit significant variability across datasets. This highlights that most defenses do not offer global protection when deployed in a scenario different from the one evaluated in their respective papers. The only defenses nearing zero are TR, I-BAU, and PDB. Examining the CA for the former two in the case of Imagenet-1K shows a complete reduction to (near) zero, while PDB completely fails with Imagenet-1K, highlighting a critical situation even in those that initially appeared more promising.

C8: Design defenses that minimize hyperparameter sensitivity across datasets, improving defense generalizability.

R9: Tune and validate defenses on multiple datasets to ensure consistent cross-dataset effectiveness.

Variability analysis Repeated experiments on CIFAR-100 with ResNet-18 under the BadNets attack (0.5% poison rate) reveal noticeable variability across defenses. The CA remains stable for most methods, while ASR shows high fluctuations on more than half

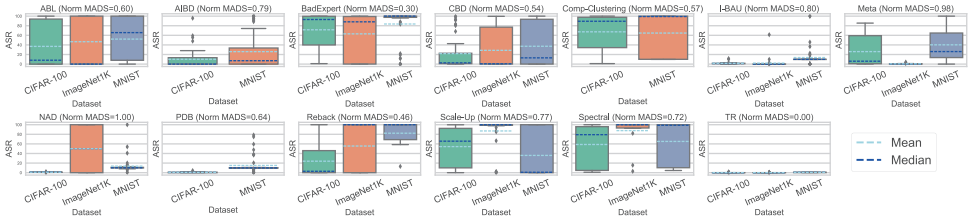


Figure 12.15: Comparison of the attack ASR on various datasets using default hyperparameters to different defenses.

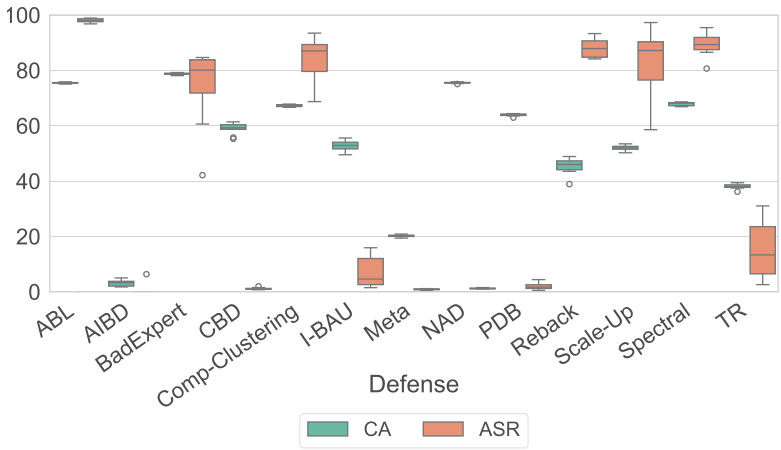


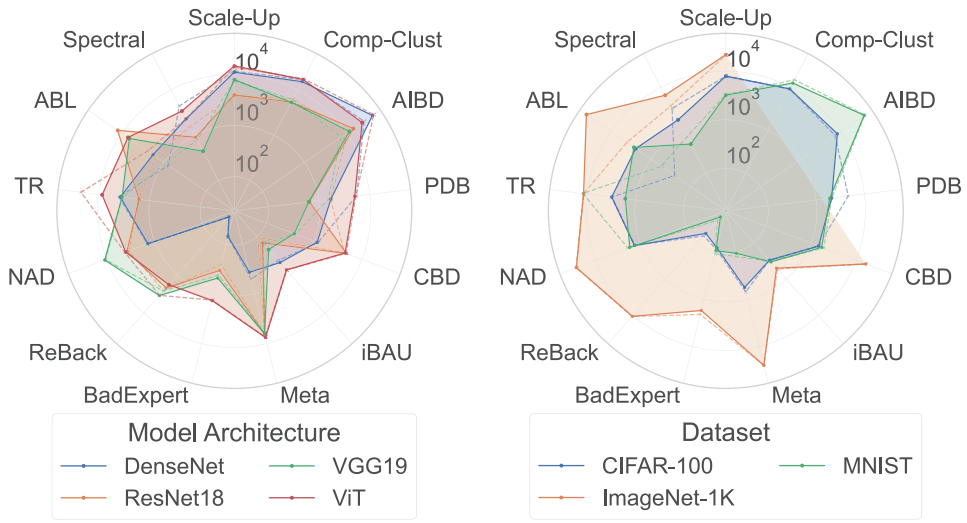
Figure 12.16: Performance consistency (CA and ASR) using BadNets against defense methods on CIFAR-100 with ResNet-18 (0.5% poison rate). Each defense shows two distributions: CA (left, green) and ASR (right, red) computed over ten independent trials.

of the evaluated defenses (see Figure 12.16). These results highlight the need to systematically measure variability and report average performance along with its dispersion (e.g., mean ± stdev, Coefficient of Variation, IQR). Reporting these results allows a precise assessment of defense stability across runs and ensures that results reflect consistent, reproducible behavior.

R10: Evaluations must include repeated runs to capture performance variability and report results using statistical measures such as the mean standard deviation (±), interquartile range (IQR), and coefficient of variation (CV).

Execution Time The analysis in Section 12.3 highlights a persistent lack of runtime reporting in defense evaluations. Execution time, while fundamental for assessing scalability and practicality, is often omitted or incompletely reported. Knowing how long a defense takes to operate, both in the presence and absence of a backdoor, is essential to evaluating its feasibility for real-world deployment.

Observations. In Figure 12.17, we report the average elapsed time of all evaluated



(a) Grouped by architecture.

(b) Grouped by dataset.

Figure 12.17: Average elapsed time (log scale) of defenses under different evaluation settings. Each radar plot summarizes the mean runtime across all attacks and poison rates. Dotted lines represent the runtime for each defense when no backdoor is present.

defenses across datasets and model architectures, comparing their runtime when a backdoor is present versus when it is not. The results show that runtime behavior is highly defense-dependent. Some methods, such as *Spectral Signatures*, *Reback*, and *CBD*, exhibit minor variation between the two conditions, indicating stable and predictable execution cost. Others, however, display substantial differences: while certain defenses run faster on clean models, others become slower without a backdoor, likely due to convergence mechanisms or additional verification when no anomalies are detected. Moreover, several defenses, specifically *Compatibility Clustering*, *AIBD*, and *PDB*, did not complete within a reasonable time frame (10 hours). These methods rely on extensive feature-space clustering or repeated gradient-based optimization procedures, which scale poorly with dataset size. These findings also further support Recommendation R6.

C9: Achieving stable and efficient runtimes across defenses remains a critical challenge to scalability and real-world deployment.

Case Study: Detection Defenses We have also tested the chosen attacks against detection-based backdoor defenses like Neural Cleanse [149], Unicorn [157], and BAN [175]. In this experiment, we poisoned 10 models and used the defenses to identify them. The results are shown in Figure 12.18. Unicorn could not be applied to single-channel (grayscale) images without modification, so we did not run experiments with Unicorn on the MNIST dataset. The results vary depending on the chosen setup. For example, successful BadNets and AdvDoor attacks are detected by defenses across different models, whereas successful

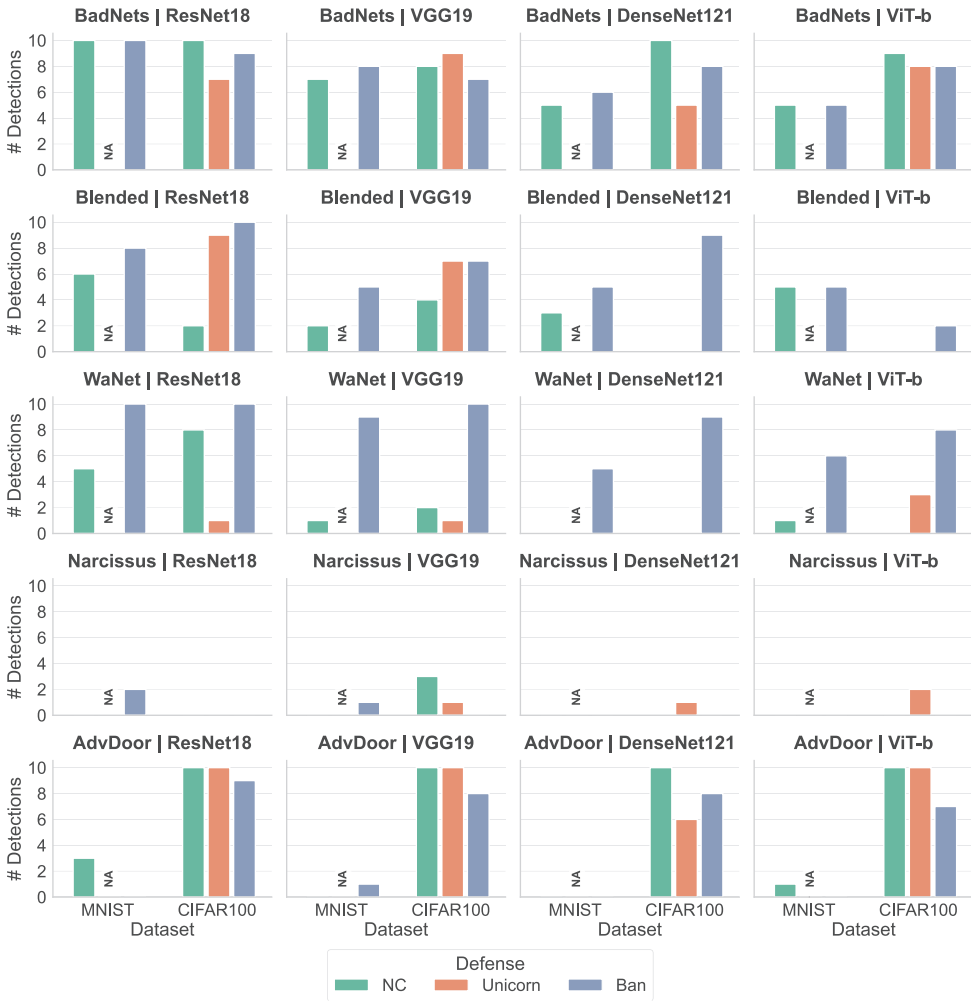


Figure 12.18: Performance of detection-based defenses on MNIST and CIFAR-100. We have no results for Unicorn and MNIST, as it could not be applied out of the box to one-dimensional images.

Narcissus evades them.[10] On average, defenses work slightly better on ResNet18 than on other model architectures. Again, the trend of evaluating only against ResNet models, BadNets attack, and simpler datasets (CIFAR-10 and MNIST) does not fully support the generalizability of the evaluated defense.

12.5.1. Discussion on Sustainability

As we reported in Section 12.5, to conduct, according to our estimates, a fair comparison among 15 defenses across more than 3 000 experiments required more than 170 days of continuous computation. While this number, reflecting only the GPU time spent on the successful and reported experiments, already represents a considerable investment of resources, the actual resources consumed were substantially greater. The total experimental time in GPU hours was significantly extended by typical large-scale evaluation overheads, including repetitions due to experimental errors and necessary adjustments. Specifically, the reported experiments required about 78% of the total available computing credits, while in reality, with the necessary implementation and preparation phase, the entire study consumed the full annual credit allocation for two accounts on the Netherlands national super-computing infrastructure (Snellius), primarily for the two large datasets, CIFAR-100 and ImageNet-1K. Several runs on Imagenet-1K frequently timed out after 10 (and 24) hours due to excessive resource consumption, while in some cases, memory consumption (over 300GB) led to failures. Moreover, experiments on MNIST were executed on an additional institutional compute cluster. This highlights serious concerns regarding the sustainability and reproducibility of current defense evaluation practices. The growing number of backdoor defenses proposed in recent years, even within A* conferences, raises serious doubts about the realism and credibility of reported evaluations. This trend reflects broader pressure to publish rapidly, often at the expense of thorough, reproducible experimentation, leading to fragmented evaluation practices and limited comparability across studies. At the same time, the lack of clear evaluation guidelines makes it difficult to design exhaustive, comparable setups, often leading researchers to rely on simple or easy-to-implement baselines and to run redundant experiments that waste valuable computational resources. These observations reveal the need to establish standardized baselines that serve as reference leaderboards, enabling researchers to compare new defenses against verified results without repeating prior experiments, thereby promoting sustainable and environmentally responsible practices.

> **R11:** Define standardized evaluation protocols to improve the efficiency and sustainability of defense evaluation.

12.6. Guidelines for Experimental Evaluation

Based on our experimental analysis, we suggest a number of guidelines to follow:

- **Diversity over quantity**. Evaluate the performance of backdoor defense with different model families (e.g., CNN and ViT), dataset complexities (MNIST, CIFAR-100, and ImageNet-1K), and attack types (trigger in input-, feature-, and parameter-

[10] Note that for MNIST, Narcissus and AdvDoor attacks were not effective (low ASR), making low detection of backdoor reasonable.

space). Compare with other state-of-the-art defenses with similar threat models or objectives.
- **Fair comparison**. Use the default hyperparameters for the new defense when comparing with defenses that use default hyperparameters. If the hyperparameters are tuned, provide the same tuning effort for each defense.
- **Result reporting**. Besides common metrics like ASR and CA, provide results on execution time and the model's behavior when defense is deployed, but no attack is present.
- **Statistical significance**. Provide multiple runs of experiments and report average and standard deviation values.
- **Adaptive attacker**. Always assess the performance of a defense in the presence of an adaptive attacker, which aligns with previous studies [12, 3], but is not always followed by the community.
- **Baselines**. Given the number of attacks and defenses proposed over the last few years, it is impossible to conduct a detailed experimental comparison (if for no other reason than computational constraints). As such, we see a need to maintain a public list of baseline results. Then, such arranged baselines must be reported in every paper. Of course, for this to be achieved, the community must agree that it is necessary, as it will require additional effort for experiments and to maintain the list.

12.7. LIMITATIONS

While we aimed to provide a detailed overview and experiments on backdoor attacks, we had to impose several constraints. First, we limit our attention to the image domain. While the percentage of works focusing on the image domain justifies our choice, it would be interesting to consider other domains and to compare domain-agnostic defenses. Next, we limit our attention to "only" nine venues. While those venues are commonly considered top-tier venues, there are more venues in the same category that we did not consider. Moreover, we do not consider journals. Finally, excellent defenses could be proposed in other venues (e.g., EuroS&P, ACSAC, or SaTML), making our selection, while diverse, still far from exhaustive.

In our experiments, we consider 15 defenses, five attacks, four poisoning rates, four models, and three datasets. While this represents a considerable number of experiments, it can still be considered cherry-picked among all possible options. Indeed, since we list 183 defenses, evaluating only 15 means we skip many potentially excellent defenses. Finally, to make our comparison as fair as possible, we do not tune the defenses' hyperparameters but use their default values. While we consider this option the only fair (and realistic) one, it may (and probably did) cause us to underestimate the potential performance of several defenses.

12.8. RELATED WORK

Several surveys have systematically reviewed the literature, covering attack strategies, defensive mechanisms, and general taxonomies [80, 31, 81, 42, 6]. While valuable, these works primarily focus on cataloging attack types and providing high-level defense cat-

Table 12.8: Scope comparison for related surveys and this SoK. Prior work provides high-level overviews of attacks and defenses, while our work systematically analyzes defense evaluation practices across multiple critical dimensions.

Paper	Metrics	Datasets	Models	Poisoning	Categorization	Trigger	Domains	Tasks	Exec. Time	Adaptive
Li et al. [80]	✓	✓	✓		✓					
Gao et al. [31]										
Bai et al. [6]	✓	✓								
Shao et al. [81]					✓					
BackdoorBench [162]	✓	✓	✓							
Our Work	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

egorizations rather than systematically analyzing how defenses are evaluated. More structured evaluation efforts have emerged recently. BackdoorBench [162] provides a unified benchmarking framework for comparing defenses under standardized settings. However, it evaluates only a small subset of defenses (12 methods) under limited threat models and does not address the broader methodological issues in defense evaluation across the literature. The rapid growth of backdoor defense research, with hundreds of papers published in recent years, makes ad-hoc comparisons increasingly difficult and highlights the need for systematic evaluation practices.

Table 12.8 compares the scope of prior surveys against our work. Existing surveys cover high-level aspects such as attack categories, trigger types, and domains, but rarely detail experimental parameters like poisoning rates, execution-time constraints, or adaptive attacker considerations. Moreover, most do not explicitly categorize defenses in a way that enables reproducible cross-paper comparisons. In contrast, our work focuses exclusively on defenses, cataloging them along multiple dimensions (e.g., metrics, datasets, models, execution time, attacker adaptiveness) to facilitate fair and transparent benchmarking. The works in Table 12.8 highlight both attacks and defenses, explaining the general trends in backdoor attacks and defenses. These papers focus on a general explanation rather than comparing/analyzing the current defenses. The increasing number of publications makes comparing defenses (in a benchmark, reproducible, and fair manner) intractable.

To our knowledge, this is the first SoK that focuses on defense evaluation in backdoor attacks. Rather than surveying attack or defense techniques, we systematically analyze how the community evaluates defenses across 183 papers from top-tier AI and security venues. We identify critical gaps in evaluation practices, including reliance on outdated attack methods, low-difficulty datasets, and weak threat models, among others, and provide actionable recommendations for rigorous, reproducible defense evaluation. Our analysis reveals that current evaluation practices may substantially overestimate defense effectiveness, with significant implications for deploying these techniques in real-world systems.

12.9. CONCLUSIONS & FUTURE WORK

This work presents the first comprehensive systematization of backdoor defense evaluation. By analyzing 183 papers and conducting over 3 000 experiments, this study offers a clear, data-driven perspective on dominant methodological trends and identifies key gaps within the existing literature. Our findings reveal that 73% of defenses evaluate only on MNIST or CIFAR-10, only 58% consider adaptive attackers, and execution time is rarely reported (≈ 5%). Empirically, we demonstrate that defense effectiveness varies dramatically across settings—methods that succeed on MNIST often fail on ImageNet-1K, and defenses robust to patch triggers struggle against more complex backdoors. We provide concrete recommendations for evaluation: prioritize dataset diversity over quantity, test against both established and recent attacks (Recommendation R3), report computational overhead (Recommendation R6), and evaluate behavior under no attack conditions (Recommendation R7). Critical open challenges remain: developing architecture-agnostic defenses (Challenge C5), achieving robustness across trigger types (Challenge C6), and designing hyperparameter-insensitive methods (Challenge C8), to name a few. Our analysis—170 GPU-days for this study—underscores the need for standardized evaluation protocols to enable sustainable, reproducible research.

By releasing our framework and dataset, we aim to facilitate fair comparisons of defenses and guide practitioners in selecting appropriate defenses for real-world deployment. Future work should focus on establishing community-maintained leaderboards and developing theoretical frameworks that connect defense mechanisms to attack characteristics, rather than focusing on empirical results.

REFERENCES

[1] Neel Alex et al. “Protecting against simultaneous data poisoning attacks”. In: *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL: https://openreview.net/forum?id=rK0YJwL69S.

[2] Shengwei An et al. “Elijah: Eliminating Backdoors Injected in Diffusion Models via Distribution Shift”. In: *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*. Ed. by Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan. AAAI Press, 2024, pp. 10847–10855. DOI: 10.1609/AAAI.V38I10.28958. URL: https://doi.org/10.1609/aaai.v38i10.28958.

[3] Daniel Arp et al. “Dos and don’ts of machine learning in computer security”. In: *31st USENIX Security Symposium (USENIX Security 22)*. 2022, pp. 3971–3988.

[4] Ahmadreza Azizi et al. “T-Miner: A Generative Approach to Defend Against Trojan Attacks on DNN-based Text Classification”. In: *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*. Ed. by Michael D. Bailey and Rachel Greenstadt. USENIX Association, 2021, pp. 2255–2272. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/azizi.

[5] Eugene Bagdasaryan and Vitaly Shmatikov. "Blind backdoors in deep learning models". In: *30th USENIX Security Symposium (USENIX Security 21)*. 2021, pp. 1505–1521.

[6] Yang Bai et al. "Backdoor attack and defense on deep learning: A survey". In: *IEEE Transactions on Computational Social Systems* (2024).

[7] Ahmed Bensaoud, Jugal Kalita, and Mahmoud Bensaoud. "A survey of malware detection using deep learning". In: *Machine Learning With Applications* 16 (2024), p. 100546.

[8] Shubham Bharti et al. "Provable defense against backdoor policies in reinforcement learning". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 14704–14714.

[9] Ruisi Cai et al. "Randomized Channel Shuffling: Minimal-Overhead Backdoor Attack Detection without Clean Datasets". In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo et al. 2022. URL: http://papers.nips.cc/paper%5C_files/paper/2022/hash/db1d5c63576587fc1d40d33a75190c71-Abstract-Conference.html.

[10] Xiaoyu Cao et al. "FedRecover: Recovering from Poisoning Attacks in Federated Learning using Historical Information". In: *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*. IEEE, 2023, pp. 1366–1383. DOI: 10.1109/SP46215.2023.10179336. URL: https://doi.org/10.1109/SP46215.2023.10179336.

[11] Xiaoyu Cao et al. "Fltrust: Byzantine-robust federated learning via trust bootstrapping". In: *arXiv preprint arXiv:2012.13995* (2020).

[12] Nicholas Carlini et al. "On evaluating adversarial robustness". In: *arXiv preprint arXiv:1902.06705* (2019).

[13] Shuwen Chai and Jinghui Chen. "One-shot Neural Backdoor Erasing via Adversarial Weight Masking". In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo et al. 2022. URL: http://papers.nips.cc/paper%5C_files/paper/2022/hash/8c0f7107ab85892ccf51f0a814957af1-Abstract-Conference.html.

[14] Weixin Chen, Baoyuan Wu, and Haoqian Wang. "Effective Backdoor Defense by Exploiting Sensitivity of Poisoned Samples". In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo et al. 2022. URL: http://papers.nips.cc/paper%5C_files/paper/2022/hash/3f9bbf77fbd858e5b6e39d39fe84ed2e-Abstract-Conference.html.

[15] Xinyun Chen et al. "Targeted backdoor attacks on deep learning systems using data poisoning". In: *arXiv preprint arXiv:1712.05526* (2017).

[16] Xuan Chen et al. “BIRD: Generalizable Backdoor Detection and Removal for Deep Reinforcement Learning”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023. URL: http://papers.nips.cc/paper%5C_files/paper/2023/hash/802e90325f4c8546e13e5763b2ecab88-Abstract-Conference.html.

[17] Yiming Chen, Haiwei Wu, and Jiantao Zhou. “Progressive Poisoned Data Isolation for Training-Time Backdoor Defense”. In: *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*. Ed. by Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan. AAAI Press, 2024, pp. 11425–11433. DOI: 10.1609/AAAI.V38I10.29023. URL: https://doi.org/10.1609/aaai.v38i10.29023.

[18] Yukun Chen et al. “REFINE: Inversion-Free Backdoor Defense via Model Reprogramming”. In: *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL: https://openreview.net/forum?id=4IYdCws9fc.

[19] Siyuan Cheng et al. “BEAGLE: Forensics of Deep Learning Backdoor Attack for Better Defense”. In: *30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*. The Internet Society, 2023. URL: https://www.ndss-symposium.org/ndss-paper/beagle-forensics-of-deep-learning-backdoor-attack-for-better-defense/.

[20] Siyuan Cheng et al. “OdScan: Backdoor Scanning for Object Detection Models”. In: *IEEE Symposium on Security and Privacy, SP 2024, San Francisco, CA, USA, May 19-23, 2024*. IEEE, 2024, pp. 1703–1721. DOI: 10.1109/SP54263.2024.00119. URL: https://doi.org/10.1109/SP54263.2024.00119.

[21] Tianyue Chu et al. “Securing Federated Sensitive Topic Classification against Poisoning Attacks”. In: *30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*. The Internet Society, 2023. URL: https://www.ndss-symposium.org/ndss-paper/securing-federated-sensitive-topic-classification-against-poisoning-attacks/.

[22] Khoa Doan et al. “LIRA: Learnable, Imperceptible and Robust Backdoor Attacks”. In: *International Conference on Computer Vision (ICCV)*. 2021.

[23] Khoa D. Doan et al. “Defending Backdoor Attacks on Vision Transformer via Patch Processing”. In: *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*. Ed. by Brian Williams, Yiling Chen, and Jennifer Neville. AAAI Press, 2023, pp. 506–515. DOI: 10.1609/AAAI.V37I1.25125. URL: https://doi.org/10.1609/aaai.v37i1.25125.

[24] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020).

[25] Min Du, Ruoxi Jia, and Dawn Song. "Robust anomaly detection and backdoor attack detection via differential privacy". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=SJx0q1rtvS.

[26] Shiwei Feng et al. "Detecting backdoors in pre-trained encoders". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 16352–16362.

[27] Hossein Fereidooni et al. "FreqFed: A Frequency Analysis-Based Approach for Mitigating Poisoning Attacks in Federated Learning". In: *31st Annual Network and Distributed System Security Symposium, NDSS 2024, San Diego, California, USA, February 26 - March 1, 2024*. The Internet Society, 2024. URL: https://www.ndss-symposium.org/ndss-paper/freqfed-a-frequency-analysis-based-approach-for-mitigating-poisoning-attacks-in-federated-learning/.

[28] Greg Fields et al. "Trojan signatures in DNN weights". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 12–20.

[29] Chong Fu et al. "FreeEagle: Detecting Complex Neural Trojans in Data-Free Cases". In: *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*. Ed. by Joseph A. Calandrino and Carmela Troncoso. USENIX Association, 2023, pp. 6399–6416. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/fu-chong.

[30] Kuofeng Gao et al. "Backdoor defense via adaptively splitting poisoned dataset". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 4005–4014.

[31] Yansong Gao et al. "Backdoor attacks and countermeasures on deep learning: A comprehensive review". In: *arXiv preprint arXiv:2007.10760* (2020).

[32] Yansong Gao et al. "STRIP: a defence against trojan attacks on deep neural networks". In: *Annual Computer Security Applications Conference (ACSAC)*. 2019.

[33] Yudong Gao et al. "Energy-based Backdoor Defense without Task-Specific Samples and Model Retraining". In: *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=TJ6tVNt6Y4.

[34] Xueluan Gong et al. "Redeem Myself: Purifying Backdoors in Deep Learning Models using Self Attention Distillation". In: *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*. IEEE, 2023, pp. 755–772. DOI: 10.1109/SP46215.2023.10179375. URL: https://doi.org/10.1109/SP46215.2023.10179375.

[35] Sorin Grigorescu et al. "A survey of deep learning techniques for autonomous driving". In: *Journal of field robotics* 37.3 (2020), pp. 362–386.

[36] Tianyu Gu et al. "BadNets: Evaluating Backdooring Attacks on Deep Neural Networks". In: *IEEE Access* (2019).

[37] Jiyang Guan, Jian Liang, and Ran He. “Backdoor defense via test-time detecting and repairing”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 24564–24573.

[38] Jiyang Guan et al. “Few-shot backdoor defense using shapley estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 13358–13367.

[39] Zihan Guan et al. “UFID: A Unified Framework for Black-box Input-level Backdoor Detection on Diffusion Models”. In: *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*. Ed. by Toby Walsh, Julie Shah, and Zico Kolter. AAAI Press, 2025, pp. 27312–27320. DOI: 10.1609/AAAI.V39I26.34941. URL: https://doi.org/10.1609/aaai.v39i26.34941.

[40] Junfeng Guo, Ang Li, and Cong Liu. “AEVA: Black-box Backdoor Detection Using Adversarial Extreme Value Analysis”. In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: https://openreview.net/forum?id=OM%5C_lYiHXiCL.

[41] Junfeng Guo et al. “SCALE-UP: An Efficient Black-box Input-level Backdoor Detection via Analyzing Scaled Prediction Consistency”. In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: https://openreview.net/forum?id=o0LFPcoFKnr.

[42] Wei Guo, Benedetta Tondi, and Mauro Barni. “An overview of backdoor attacks against deep neural networks and possible defences”. In: *IEEE Open Journal of Signal Processing* 3 (2022), pp. 261–287.

[43] Philemon Hailemariam and Birhanu Eshete. “PoisonSpot: Precise Spotting of Clean-Label Backdoors via Fine-Grained Training Provenance Tracking”. In: *Proceedings of the 2025 on ACM SIGSAC Conference on Computer and Communications Security*. CCS ’25. Taipei, Taiwan, 2025.

[44] Jonathan Hayase et al. “SPECTRE: Defending Against Backdoor Attacks Using Robust Statistics”. In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2021*. Proceedings of Machine Learning Research. PMLR, 2021.

[45] Bowei He et al. “Certifying Language Model Robustness with Fuzzed Randomized Smoothing: An Efficient Defense Against Backdoor Attacks”. In: *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL: https://openreview.net/forum?id=USI3ZbuFaV.

[46] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[47] Shuo He et al. “A Closer Look at Backdoor Attacks on CLIP”. In: *Proceedings of the 42nd International Conference on Machine Learning*. Ed. by Aarti Singh et al. Vol. 267. Proceedings of Machine Learning Research. PMLR, July 2025, pp. 22836–22852. URL: https://proceedings.mlr.press/v267/he25v.html.

[48] Wenwen He et al. “SPMC: Self-Purifying Federated Backdoor Defense via Margin Contribution”. In: *Proceedings of the 42nd International Conference on Machine Learning*. Ed. by Aarti Singh et al. Vol. 267. Proceedings of Machine Learning Research. PMLR, July 2025, pp. 22422–22433. URL: https://proceedings.mlr.press/v267/he25f.html.

[49] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. “Handcrafted backdoors in deep neural networks”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 8068–8080.

[50] Linshan Hou et al. “IBD-PSC: Input-level Backdoor Detection via Parameter-oriented Scaling Consistency”. In: *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=YCzbfs2few.

[51] Sizai Hou, Songze Li, and Duanyi Yao. “DeDe: Detecting Backdoor Samples for SSL Encoders via Decoders”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*. Computer Vision Foundation / IEEE, 2025, pp. 20675–20684. DOI: 10.1109/CVPR52734.2025.01925. URL: https://openaccess.thecvf.com/content/CVPR2025/html/Hou%5C_DeDe%5C_Detecting%5C_Backdoor%5C_Samples%5C_for%5C_SSL%5C_Encoders%5C_via%5C_Decoders%5C_CVPR%5C_2025%5C_paper.html.

[52] Xiaoling Hu et al. “Trigger Hunting with a Topological Prior for Trojan Detection”. In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: https://openreview.net/forum?id=TXsjU8BaibT.

[53] Dong Huang and Qingwen Bu. “Adversarial Feature Map Pruning for Backdoor”. In: *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=IOEEDkla96.

[54] Gao Huang et al. “Densely connected convolutional networks”. In: *CVPR*. 2017.

[55] Hanxun Huang et al. “Detecting Backdoor Samples in Contrastive Language Image Pretraining”. In: *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL: https://openreview.net/forum?id=KmQEsIfhr9.

[56] Kunzhe Huang et al. “Backdoor Defense via Decoupling the Training Process”. In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: https://openreview.net/forum?id=TySnJ-0RdKI.

[57] Tiansheng Huang et al. “Lockdown: Backdoor Defense for Federated Learning with Isolated Subspace Training”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023. URL: http://papers.nips.cc/paper%5C_files/paper/2023/hash/2376f25ef1725a9e3516ee3c86a59f46-Abstract-Conference.html.

[58] Tiansheng Huang et al. “Lockdown: backdoor defense for federated learning with isolated subspace training”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 10876–10896.

[59] Wenke Huang et al. “Parameter Disparities Dissection for Backdoor Defense in Heterogeneous Federated Learning”. In: *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. Ed. by Amir Globersons et al. 2024. URL: http://papers.nips.cc/paper%5C_files/paper/2024/hash/db0ee27cb50dd9087b133f6e7d28a90e-Abstract-Conference.html.

[60] Matthew Jagielski et al. “High accuracy and high fidelity extraction of neural networks”. In: *29th USENIX security symposium (USENIX Security 20)*. 2020, pp. 1345–1362.

[61] Matthew Jagielski et al. “Manipulating machine learning: Poisoning attacks and countermeasures for regression learning”. In: *2018 IEEE symposium on security and privacy (SP)*. IEEE. 2018, pp. 19–35.

[62] Jinyuan Jia et al. “Certified Robustness of Nearest Neighbors against Data Poisoning and Backdoor Attacks”. In: *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*. AAAI Press, 2022, pp. 9575–9583. DOI: 10.1609/AAAI.V36I9.21191. URL: https://doi.org/10.1609/aaai.v36i9.21191.

[63] Jinyuan Jia et al. “FedGame: A Game-Theoretic Defense against Backdoor Attacks in Federated Learning”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023. URL: http://papers.nips.cc/paper%5C_files/paper/2023/hash/a6678e2be4ce7aef9d2192e03cd586b7-Abstract-Conference.html.

[64] Charles Jin, Melinda Sun, and Martin C. Rinard. “Incompatibility Clustering as a Defense Against Backdoor Poisoning Attacks”. In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: https://openreview.net/forum?id=mkJm5Uy4HrQ.

[65] Ehsanul Kabir et al. “FLShield: A Validation Based Federated Learning Framework to Defend Against Poisoning Attacks”. In: *IEEE Symposium on Security and Privacy, SP 2024, San Francisco, CA, USA, May 19-23, 2024*. IEEE, 2024, pp. 2572–2590. DOI: 10.1109/SP54263.2024.00141. URL: https://doi.org/10.1109/SP54263.2024.00141.

[66] Alaa Khaddaj et al. “Rethinking Backdoor Attacks”. In: *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 16216–16236. URL: https://proceedings.mlr.press/v202/khaddaj23a.html.

[67] Soheil Kolouri et al. “Universal litmus patterns: Revealing backdoor attacks in cnns”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 301–310.

[68] Torsten Krauß, Jasper Stang, and Alexandra Dmitrienko. “Verify your Labels! Trustworthy Predictions and Datasets via Confidence Scores”. In: *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*. Ed. by Davide Balzarotti and Wenyuan Xu. USENIX Association, 2024. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/kraub-verify.

[69] Kavita Kumari et al. “BayBFed: Bayesian Backdoor Defense for Federated Learning”. In: *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*. IEEE, 2023, pp. 737–754. DOI: 10.1109/SP46215.2023.10179362. URL: https://doi.org/10.1109/SP46215.2023.10179362.

[70] Simon Lachnit and Ghassan Karame. “On Hyperparameters and Backdoor-Resistance in Horizontal Federated Learning”. In: *Proceedings of the 2025 on ACM SIGSAC Conference on Computer and Communications Security*. CCS ’25. Taipei, Taiwan, 2025.

[71] Alexander Levine and Soheil Feizi. “Deep Partition Aggregation: Provable Defenses against General Poisoning Attacks”. In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=YUGG2tFuPM.

[72] Boheng Li et al. “Nearest is not dearest: Towards practical defense against quantization-conditioned backdoor attacks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 24523–24533.

[73] Boheng Li et al. “Purifying Quantization-conditioned Backdoors via Layer-wise Activation Correction with Distribution Approximation”. In: *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=CEfr3h68KU.

[74] Haoran Li et al. “Simulate and Eliminate: Revoke Backdoors for Generative Large Language Models”. In: *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*. Ed. by Toby Walsh, Julie Shah, and Zico Kolter. AAAI Press, 2025, pp. 397–405. DOI: 10.1609/AAAI.V39I1.32018. URL: https://doi.org/10.1609/aaai.v39i1.32018.

[75] Songze Li and Yanbo Dai. “BackdoorIndicator: Leveraging OOD Data for Proactive Backdoor Detection in Federated Learning”. In: *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*. Ed. by Davide Balzarotti and Wenyuan Xu. USENIX Association, 2024. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/li-songze.

[76] Wei Li et al. “PSBD: Prediction shift uncertainty unlocks backdoor detection”. In: *Proceedings of the Computer Vision and Pattern Recognition Conference*. 2025, pp. 10255–10264.

[77] Yige Li et al. “Anti-Backdoor Learning: Training Clean Models on Poisoned Data”. In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. Ed. by Marc’Aurelio Ranzato et al. 2021, pp. 14900–14912. URL: https://proceedings.neurips.cc/paper/2021/hash/7d38b1e9bd793d3f45e0e212a729a93c-Abstract.html.

[78] Yige Li et al. “Neural Attention Distillation: Erasing Backdoor Triggers from Deep Neural Networks”. In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=9l0K4OM-oXE.

[79] Yige Li et al. “Reconstructive Neuron Pruning for Backdoor Defense”. In: *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 19837–19854. URL: https://proceedings.mlr.press/v202/li23v.html.

[80] Yiming Li et al. *Backdoor Learning: A Survey*. 2022. arXiv: 2007.08745 [cs.CR].

[81] Yudong Li et al. “Backdoor attacks to deep learning models and countermeasures: A survey”. In: *IEEE Open Journal of the Computer Society* 4 (2023), pp. 134–146.

[82] Weilin Lin et al. “Fusing Pruned and Backdoored Models: Optimal Transport-based Data-free Backdoor Mitigation”. In: *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*. Ed. by Toby Walsh, Julie Shah, and Zico Kolter. AAAI Press, 2025, pp. 26299–26307. DOI: 10.1609/AAAI.V39I25.34828. URL: https://doi.org/10.1609/aaai.v39i25.34828.

[83] Weilin Lin et al. “Unveiling and Mitigating Backdoor Vulnerabilities based on Unlearning Weight Changes and Backdoor Activeness”. In: *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. Ed. by Amir Globersons et al. 2024. URL: http://papers.nips.cc/paper%5C_files/paper/2024/hash/4a3ffc1a0074901718ab9dcef41909e9-Abstract-Conference.html.

[84] Hongbin Liu, Michael K. Reiter, and Neil Zhenqiang Gong. “Mudjacking: Patching Backdoor Vulnerabilities in Foundation Models”. In: *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*. Ed. by Davide Balzarotti and Wenyuan Xu. USENIX Association, 2024. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/liu-hongbin.

[85] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. “Fine-pruning: Defending against backdooring attacks on deep neural networks”. In: *Research in Attacks, Intrusions, and Defenses (RAID)*. Springer. 2018, pp. 273–294.

[86] Shijie Liu et al. “Enhancing the Antidote: Improved Pointwise Certifications against Poisoning Attacks”. In: *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*. Ed. by Brian Williams, Yiling Chen, and Jennifer Neville. AAAI Press, 2023, pp. 8861–8869. DOI: 10.1609/AAAI.V37I7.26065. URL: https://doi.org/10.1609/aaai.v37i7.26065.

[87] Tian Yu Liu, Yu Yang, and Baharan Mirzasoleiman. “Friendly Noise against Adversarial Noise: A Powerful Defense against Data Poisoning Attack”. In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo et al. 2022. URL: http://papers.nips.cc/paper%5C_files/paper/2022/hash/4e81308aa2eb8e2e4eccf122d4827af7-Abstract-Conference.html.

[88] Xiaogeng Liu et al. “Detecting backdoors during the inference stage based on corruption robustness consistency”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 16363–16372.

[89] Yingqi Liu et al. “Abs: Scanning neural networks for back-doors by artificial brain stimulation”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019, pp. 1265–1282.

[90] Yingqi Liu et al. “Complex backdoor detection by symmetric feature differencing”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 15003–15013.

[91] Yiran Liu et al. “Causality Based Front-door Defense Against Backdoor Attack on Language Models”. In: *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=dmHHVcHFdM.

[92] Wanlun Ma et al. “The "Beatrix" Resurrections: Robust Backdoor Detection via Gram Matrices”. In: *30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*. The Internet Society, 2023. URL: https://www.ndss-symposium.org/ndss-paper/the-beatrix-resurrections-robust-backdoor-detection-via-gram-matrices/.

[93] Zhuo Ma et al. “Need for Speed: Taming Backdoor Attacks with Speed and Precision”. In: *IEEE Symposium on Security and Privacy, SP 2024, San Francisco, CA, USA, May 19-23, 2024*. IEEE, 2024, pp. 1217–1235. DOI: 10.1109/SP54263.2024.00216. URL: https://doi.org/10.1109/SP54263.2024.00216.

[94] Peihua Mai, Ran Yan, and Yan Pang. “RFLPA: A Robust Federated Learning Framework against Poisoning Attacks with Secure Aggregation”. In: *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. Ed. by Amir Globersons et al. 2024. URL: http://papers.nips.cc/paper%5C_files/paper/2024/hash/bcbdc25dc4f0be5ae8ac07232df6e33a-Abstract-Conference.html.

[95] Nay Myat Min et al. “CROW: Eliminating Backdoors from Large Language Models via Internal Consistency Regularization”. In: *Proceedings of the 42nd International Conference on Machine Learning*. Ed. by Aarti Singh et al. Vol. 267. Proceedings of Machine Learning Research. PMLR, July 2025, pp. 44272–44291. URL: https://proceedings.mlr.press/v267/min25b.html.

[96] Rui Min et al. “Towards Stable Backdoor Purification through Feature Shift Tuning”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023. URL: http://papers.nips.cc/paper%5C_files/paper/2023/hash/ee37d51b3c003d89acba2363dde256af-Abstract-Conference.html.

[97] Rui Min et al. “Uncovering, Explaining, and Mitigating the Superficial Safety of Backdoor Defense”. In: *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. Ed. by Amir Globersons et al. 2024. URL: http://papers.nips.cc/paper%5C_files/paper/2024/hash/8e8399e5e7aed601c9f135f40be26564-Abstract-Conference.html.

[98] Hossein Mirzaei et al. “Scanning Trojaned Models Using Out-of-Distribution Samples”. In: *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. Ed. by Amir Globersons et al. 2024. URL: http://papers.nips.cc/paper%5C_files/paper/2024/hash/ef52fd1e24634cb8f7003ebbfb3644d9-Abstract-Conference.html.

[99] Xiaoxing Mo et al. “Robust Backdoor Detection for Deep Learning via Topological Evolution Dynamics”. In: *IEEE Symposium on Security and Privacy, SP 2024, San Francisco, CA, USA, May 19-23, 2024*. IEEE, 2024, pp. 2048–2066. DOI: 10.1109/SP54263.2024.00174. URL: https://doi.org/10.1109/SP54263.2024.00174.

[100] Yichuan Mo et al. “TERD: A Unified Framework for Safeguarding Diffusion Models Against Backdoors”. In: *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=lpHjmPvxW1.

[101] Bingxu Mu et al. “Progressive backdoor erasing via connecting backdoor and adversarial attacks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 20495–20503.

[102] Dung Thuy Nguyen et al. “PBP: Post-training Backdoor Purification for Malware Classifiers”. In: *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society, 2025. URL: https://www.ndss-symposium.org/ndss-paper/pbp-post-training-backdoor-purification-for-malware-classifiers/.

[103] Thien Duc Nguyen et al. “FLAME: Taming Backdoors in Federated Learning”. In: *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*. Ed. by Kevin R. B. Butler and Kurt Thomas. USENIX Association, 2022,

pp. 1415–1432. URL: https://www.usenix.org/conference/usenixsecurity22/presentation/nguyen.

[104] Tuan Anh Nguyen and Anh Tuan Tran. "WaNet - Imperceptible Warping-based Backdoor Attack". In: *ICLR*. 2021.

[105] Yuwei Niu et al. "Test-Time Multimodal Backdoor Detection by Contrastive Prompting". In: *Proceedings of the 42nd International Conference on Machine Learning*. Ed. by Aarti Singh et al. Vol. 267. Proceedings of Machine Learning Research. PMLR, July 2025, pp. 46629–46648. URL: https://proceedings.mlr.press/v267/niu25b.html.

[106] Mustafa Safa Özdayi, Murat Kantarcioglu, and Yulia R. Gel. "Defending against Backdoors in Federated Learning with Robust Learning Rate". In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 9268–9276. DOI: 10.1609/AAAI.V35I10.17118. URL: https://doi.org/10.1609/aaai.v35i10.17118.

[107] Soumyadeep Pal et al. "Backdoor Secrets Unveiled: Identifying Backdoor Data with Optimized Scaled Prediction Consistency". In: *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=1OfAO2mes1.

[108] Minzhou Pan et al. "ASSET: Robust Backdoor Data Detection Across a Multiplicity of Deep Learning Paradigms". In: *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*. Ed. by Joseph A. Calandrino and Carmela Troncoso. USENIX Association, 2023, pp. 2725–2742. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/pan.

[109] Zhuoshi Pan et al. "From Trojan Horses to Castle Walls: Unveiling Bilateral Data Poisoning Effects in Diffusion Models". In: *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. Ed. by Amir Globersons et al. 2024. URL: http://papers.nips.cc/paper%5C_files/paper/2024/hash/95dcc1f6463491d37a8918c1d38380a7-Abstract-Conference.html.

[110] Lu Pang et al. "Backdoor cleansing with unlabeled data". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 12218–12227.

[111] Hengzhi Pei et al. "TextGuard: Provable Defense against Backdoor Attacks on Text Classification". In: *31st Annual Network and Distributed System Security Symposium, NDSS 2024, San Diego, California, USA, February 26 - March 1, 2024*. The Internet Society, 2024. URL: https://www.ndss-symposium.org/ndss-paper/textguard-provable-defense-against-backdoor-attacks-on-text-classification/.

[112] Jie Peng et al. “Circumventing Backdoor Space via Weight Symmetry”. In: *Proceedings of the 42nd International Conference on Machine Learning*. Ed. by Aarti Singh et al. Vol. 267. Proceedings of Machine Learning Research. PMLR, July 2025, pp. 48886–48922. URL: https://proceedings.mlr.press/v267/peng25f.html.

[113] Omead Pooladzandi et al. “PureGen: Universal Data Purification for Train-Time Poison Defense via Generative Model Dynamics”. In: *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. Ed. by Amir Globersons et al. 2024. URL: http://papers.nips.cc/paper%5C_files/paper/2024/hash/f4757db82a02eea015670ecca605d5cc-Abstract-Conference.html.

[114] Dorde Popovic et al. “DeBackdoor: A Deductive Framework for Detecting Backdoor Attacks on Deep Models with Limited Data”. In: *34th USENIX Security Symposium, USENIX Security 2025, Seattle, WA, USA, August 13-15, 2025*. Ed. by Lujo Bauer and Giancarlo Pellegrino. USENIX Association, 2025, pp. 6419–6438. URL: https://www.usenix.org/conference/usenixsecurity25/presentation/popovic.

[115] Xiangyu Qi et al. “Towards A Proactive ML Approach for Detecting Backdoor Poison Samples”. In: *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*. Ed. by Joseph A. Calandrino and Carmela Troncoso. USENIX Association, 2023, pp. 1685–1702. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/qi.

[116] Ximing Qiao, Yukun Yang, and Hai Li. “Defending Neural Backdoors via Generative Distribution Modeling”. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al. 2019, pp. 14004–14013. URL: https://proceedings.neurips.cc/paper/2019/hash/78211247db84d96acf4e00092a7fba80-Abstract.html.

[117] Zhen Qin et al. “Resisting Backdoor Attacks in Federated Learning via Bidirectional Elections and Individual Perspective”. In: *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*. Ed. by Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan. AAAI Press, 2024, pp. 14677–14685. DOI: 10.1609/AAAI.V38I13.29385. URL: https://doi.org/10.1609/aaai.v38i13.29385.

[118] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. “Tbt: Targeted neural network attack with bit trojan”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 13198–13207.

[119] Zhiyao Ren et al. “ICLShield: Exploring and Mitigating In-Context Learning Backdoor Attacks”. In: *Proceedings of the 42nd International Conference on Machine Learning*. Ed. by Aarti Singh et al. Vol. 267. Proceedings of Machine Learning Research. PMLR, July 2025, pp. 51436–51449. URL: https://proceedings.mlr.press/v267/ren25b.html.

[120] Phillip Rieger et al. "CrowdGuard: Federated Backdoor Detection in Federated Learning". In: *31st Annual Network and Distributed System Security Symposium, NDSS 2024, San Diego, California, USA, February 26 - March 1, 2024*. The Internet Society, 2024. URL: https://www.ndss-symposium.org/ndss-paper/crowdguard-federated-backdoor-detection-in-federated-learning/.

[121] Phillip Rieger et al. "DeepSight: Mitigating Backdoor Attacks in Federated Learning Through Deep Model Inspection". In: *29th Annual Network and Distributed System Security Symposium, NDSS 2022, San Diego, California, USA, April 24-28, 2022*. The Internet Society, 2022. URL: https://www.ndss-symposium.org/ndss-paper/auto-draft-205/.

[122] Phillip Rieger et al. "SafeSplit: A Novel Defense Against Client-Side Backdoor Attacks in Split Learning". In: *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society, 2025. URL: https://www.ndss-symposium.org/ndss-paper/safesplit-a-novel-defense-against-client-side-backdoor-attacks-in-split-learning/.

[123] Elan Rosenfeld et al. "Certified Robustness to Label-Flipping Attacks via Randomized Smoothing". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 8230–8241. URL: http://proceedings.mlr.press/v119/rosenfeld20b.html.

[124] Giorgio Severi et al. "{Explanation-Guided} backdoor poisoning attacks against malware classifiers". In: *30th USENIX security symposium (USENIX security 21)*. 2021, pp. 1487–1504.

[125] Shawn Shan et al. "Poison Forensics: Traceback of Data Poisoning Attacks in Neural Networks". In: *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*. Ed. by Kevin R. B. Butler and Kurt Thomas. USENIX Association, 2022, pp. 3575–3592. URL: https://www.usenix.org/conference/usenixsecurity22/presentation/shan.

[126] Uday Shankar Shanthamallu, Jayaraman J. Thiagarajan, and Andreas Spanias. "Uncertainty-Matching Graph Neural Networks to Defend Against Poisoning Attacks". In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 9524–9532. DOI: 10.1609/AAAI.V35I11.17147. URL: https://doi.org/10.1609/aaai.v35i11.17147.

[127] Virat Shejwalkar and Amir Houmansadr. "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning". In: *NDSS*. 2021.

[128] Guangyu Shen et al. "Backdoor Scanning for Deep Neural Networks through K-Arm Optimization". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 9525–9536. URL: http://proceedings.mlr.press/v139/shen21c.html.

[129] Guangyu Shen et al. “BAIT: Large Language Model Backdoor Scanning by Inverting Attack Target”. In: *IEEE Symposium on Security and Privacy, SP 2025, San Francisco, CA, USA, May 12-15, 2025*. Ed. by Marina Blanton, William Enck, and Cristina Nita-Rotaru. IEEE, 2025, pp. 1676–1694. DOI: 10.1109/SP61157.2025.00103. URL: https://doi.org/10.1109/SP61157.2025.00103.

[130] Guangyu Shen et al. “Constrained Optimization with Dynamic Bound-scaling for Effective NLP Backdoor Defense”. In: *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*. Ed. by Kamalika Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 19879–19892. URL: https://proceedings.mlr.press/v162/shen22e.html.

[131] Guangyu Shen et al. “Django: Detecting Trojans in Object Detection Models via Gaussian Focus Calibration”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023. URL: http://papers.nips.cc/paper%5C_files/paper/2023/hash/a102d6cb996be3482c059c1e18bbe523-Abstract-Conference.html.

[132] Yangyang Shen et al. “Bi-perspective Splitting Defense: Achieving Clean-Seed-Free Backdoor Security”. In: *Proceedings of the 42nd International Conference on Machine Learning*. Ed. by Aarti Singh et al. Vol. 267. Proceedings of Machine Learning Research. PMLR, July 2025, pp. 54431–54456. URL: https://proceedings.mlr.press/v267/shen25d.html.

[133] Yucheng Shi et al. “Black-box Backdoor Defense via Zero-shot Image Purification”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023. URL: http://papers.nips.cc/paper%5C_files/paper/2023/hash/b36554b97da741b1c48c9de05c73993e-Abstract-Conference.html.

[134] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[135] Bing Sun et al. “Neural Network Semantic Backdoor Detection and Mitigation: A Causality-Based Approach”. In: *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*. Ed. by Davide Balzarotti and Wenyuan Xu. USENIX Association, 2024. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/sun-bing.

[136] Jingwei Sun et al. “FL-WBC: Enhancing Robustness against Model Poisoning Attacks in Federated Learning from a Client Perspective”. In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. Ed. by Marc’Aurelio Ranzato et al. 2021, pp. 12613–12624. URL: https://proceedings.neurips.cc/paper/2021/hash/692baebec3bb4b53d7ebc3b9fabac31b-Abstract.html.

[137] Mingjie Sun and Zico Kolter. “Single image backdoor inversion via robust smoothed classifiers”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 8113–8122.

[138] Xiaofei Sun et al. “Defending against Backdoor Attacks in Natural Language Generation”. In: *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*. Ed. by Brian Williams, Yiling Chen, and Jennifer Neville. AAAI Press, 2023, pp. 5257–5265. DOI: 10.1609/AAAI.V37I4.25656. URL: https://doi.org/10.1609/aaai.v37i4.25656.

[139] Zhen Sun et al. “PEFTGuard: Detecting Backdoor Attacks Against Parameter-Efficient Fine-Tuning”. In: *IEEE Symposium on Security and Privacy, SP 2025, San Francisco, CA, USA, May 12-15, 2025*. Ed. by Marina Blanton, William Enck, and Cristina Nita-Rotaru. IEEE, 2025, pp. 1713–1731. DOI: 10.1109/SP61157.2025.00161. URL: https://doi.org/10.1109/SP61157.2025.00161.

[140] Christian Szegedy et al. “Intriguing properties of neural networks”. In: *arXiv preprint arXiv:1312.6199* (2013).

[141] Di Tang et al. “Demon in the Variant: Statistical Analysis of DNNs for Robust Backdoor Contamination Detection”. In: *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*. Ed. by Michael D. Bailey and Rachel Greenstadt. USENIX Association, 2021, pp. 1541–1558. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/tang-di.

[142] Ruixiang (Ryan) Tang et al. “Setting the Trap: Capturing and Defeating Backdoors in Pretrained Language Models through Honeypots”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023. URL: http://papers.nips.cc/paper%5C_files/paper/2023/hash/e7938ede51225b490bb69f7b361a9259-Abstract-Conference.html.

[143] Guanhong Tao et al. “Better trigger inversion optimization in backdoor scanning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 13368–13378.

[144] Ajinkya Tejankar et al. “Defending against patch-based backdoor attacks on self-supervised learning”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 12239–12249.

[145] Brandon Tran, Jerry Li, and Aleksander Madry. “Spectral Signatures in Backdoor Attacks”. In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio et al. 2018, pp. 8011–8021. URL: https://proceedings.neurips.cc/paper/2018/hash/280cf18baf4311c92aa5a042336587d3-Abstract.html.

[146] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. “Label-consistent backdoor attacks”. In: *arXiv preprint arXiv:1912.02771* (2019).

[147] Guancheng Wan et al. “Energy-based Backdoor Defense Against Federated Graph Learning”. In: *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL: https://openreview.net/forum?id=5Jc7r5aqHJ.

[148] Bolun Wang et al. “Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks”. In: *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*. IEEE, 2019, pp. 707–723. DOI: 10.1109/SP.2019.00031. URL: https://doi.org/10.1109/SP.2019.00031.

[149] Bolun Wang et al. “Neural cleanse: Identifying and mitigating backdoor attacks in neural networks”. In: *S&P*. 2019.

[150] Hang Wang et al. “MM-BD: Post-Training Detection of Backdoor Attacks with Arbitrary Backdoor Pattern Types Using a Maximum Margin Statistic”. In: *IEEE Symposium on Security and Privacy, SP 2024, San Francisco, CA, USA, May 19-23, 2024*. IEEE, 2024, pp. 1994–2012. DOI: 10.1109/SP54263.2024.00015. URL: https://doi.org/10.1109/SP54263.2024.00015.

[151] Haotao Wang et al. “Trap and Replace: Defending Backdoor Attacks by Trapping Them into an Easy-to-Replace Subnetwork”. In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo et al. 2022. URL: http://papers.nips.cc/paper%5C_files/paper/2022/hash/ea06e6e9e80f1c3d382317fff67041ac-Abstract-Conference.html.

[152] Jialai Wang et al. “Aegis: Mitigating Targeted Bit-flip Attacks against Deep Neural Networks”. In: *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*. Ed. by Joseph A. Calandrino and Carmela Troncoso. USENIX Association, 2023, pp. 2329–2346. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/wang-jialai.

[153] Tong Wang et al. “Inspecting Prediction Confidence for Detecting Black-Box Backdoor Attacks”. In: *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*. Ed. by Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan. AAAI Press, 2024, pp. 274–282. DOI: 10.1609/AAAI.V38I1.27780. URL: https://doi.org/10.1609/aaai.v38i1.27780.

[154] Wenxiao Wang and Soheil Feizi. “Temporal Robustness against Data poisoning”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023. URL: http://papers.nips.cc/paper%5C_files/paper/2023/hash/94bcb01789fccf15afe2764d8fe0f40e-Abstract-Conference.html.

[155] Zhenting Wang et al. “Rethinking the Reverse-engineering of Trojan Triggers”. In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo et al. 2022. URL: http://papers.nips.cc/paper%5C_files/paper/2022/hash/3f9bf45ea04c98ad7cb857f951f499e2-Abstract-Conference.html.

[156] Zhenting Wang et al. “Training with More Confidence: Mitigating Injected and Natural Backdoors During Training”. In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo et al. 2022. URL: http://papers.nips.cc/paper%5C_files/paper/2022/hash/ec0c9ca85b4ea49c7ebfb503cf55f2ae-Abstract-Conference.html.

[157] Zhenting Wang et al. “UNICORN: A Unified Backdoor Trigger Inversion Framework”. In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: https://openreview.net/forum?id=Mj7K4lglGyj.

[158] Maurice Weber et al. “RAB: Provable Robustness Against Backdoor Attacks”. In: *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*. IEEE, 2023, pp. 1311–1328. DOI: 10.1109/SP46215.2023.10179451. URL: https://doi.org/10.1109/SP46215.2023.10179451.

[159] Chengkun Wei et al. “LMSanitator: Defending Prompt-Tuning Against Task-Agnostic Backdoors”. In: *31st Annual Network and Distributed System Security Symposium, NDSS 2024, San Diego, California, USA, February 26 - March 1, 2024*. The Internet Society, 2024. URL: https://www.ndss-symposium.org/ndss-paper/lmsanitator-defending-prompt-tuning-against-task-agnostic-backdoors/.

[160] Shaokui Wei, Hongyuan Zha, and Baoyuan Wu. “Mitigating Backdoor Attack by Injecting Proactive Defensive Backdoor”. In: *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. Ed. by Amir Globersons et al. 2024. URL: http://papers.nips.cc/paper%5C_files/paper/2024/hash/9374af323abb65ce551168d44b09ad5f-Abstract-Conference.html.

[161] Shaokui Wei et al. “Shared Adversarial Unlearning: Backdoor Mitigation by Unlearning Shared Adversarial Examples”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023. URL: http://papers.nips.cc/paper%5C_files/paper/2023/hash/520425a5a4c2fb7f7fc345078b188201-Abstract-Conference.html.

[162] Baoyuan Wu et al. “BackdoorBench: A Comprehensive Benchmark of Backdoor Learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022.

[163] Dongxian Wu and Yisen Wang. “Adversarial Neuron Pruning Purifies Backdoored Deep Models”. In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. Ed. by Marc’Aurelio Ranzato et al. 2021, pp. 16913–16925. URL: https://proceedings.neurips.cc/paper/2021/hash/8cbe9ce23f42628c98f80fa0fac8b19a-Abstract.html.

[164] Zhaohan Xi et al. “Defending Pre-trained Language Models as Few-shot Learners against Backdoor Attacks”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023.

URL: http://papers.nips.cc/paper%5C_files/paper/2023/hash/677c8dc72c99482507323f313faf4738-Abstract-Conference.html.

[165] Xun Xian et al. “A Unified Detection Framework for Inference-Stage Backdoor Defenses”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023. URL: http://papers.nips.cc/paper%5C_files/paper/2023/hash/1868a3c73d0d2a44c42458575fa8514c-Abstract-Conference.html.

[166] Zhen Xiang, David J. Miller, and George Kesidis. “Post-Training Detection of Backdoor Attacks for Two-Class and Multi-Attack Scenarios”. In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: https://openreview.net/forum?id=MSgB8D4Hy51.

[167] Zhen Xiang, Zidi Xiong, and Bo Li. “CBD: A Certified Backdoor Detector Based on Local Dominant Probability”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023. URL: http://papers.nips.cc/paper%5C_files/paper/2023/hash/0fbf046448d7eea18b982001320b9a10-Abstract-Conference.html.

[168] Zhen Xiang, Zidi Xiong, and Bo Li. “UMD: Unsupervised Model Detection for X2X Backdoor Attacks”. In: *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 38013–38038. URL: https://proceedings.mlr.press/v202/xiang23a.html.

[169] Chulin Xie et al. “CRFL: Certifiably Robust Federated Learning against Backdoor Attacks”. In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 11372–11382. URL: http://proceedings.mlr.press/v139/xie21a.html.

[170] Chulin Xie et al. “Unraveling the connections between privacy and certified robustness in federated learning against poisoning attacks”. In: *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 2023, pp. 1511–1525.

[171] Tinghao Xie et al. “BaDExpert: Extracting Backdoor Functionality for Accurate Backdoor Input Detection”. In: *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=s56xikpD92.

[172] Yueqi Xie, Minghong Fang, and Neil Zhenqiang Gong. “FedREDefense: Defending against Model Poisoning Attacks for Federated Learning using Model Update Reconstruction Error”. In: *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=Wjq2bS7fTK.

[173] Jiahao Xu, Zikai Zhang, and Rui Hu. "Detecting Backdoor Attacks in Federated Learning via Direction Alignment Inspection". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*. Computer Vision Foundation / IEEE, 2025, pp. 20654–20664. DOI: 10.1109/CVPR52734.2025.01923. URL: https://openaccess.thecvf.com/content/CVPR2025/html/Xu%5C_Detecting%5C_Backdoor%5C_Attacks%5C_in%5C_Federated%5C_Learning%5C_via%5C_Direction%5C_Alignment%5C_Inspection%5C_CVPR%5C_2025%5C_paper.html.

[174] Runhua Xu et al. "Dual Defense: Enhancing Privacy and Mitigating Poisoning Attacks in Federated Learning". In: *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. Ed. by Amir Globersons et al. 2024. URL: http://papers.nips.cc/paper%5C_files/paper/2024/hash/824f98c4e9bb301b0b96fa2ae071360b-Abstract-Conference.html.

[175] Xiaoyun Xu et al. "BAN: Detecting Backdoors Activated by Adversarial Neuron Noise". In: *arXiv preprint arXiv:2405.19928* (2024).

[176] Xiaoyun Xu et al. "BAN: Detecting Backdoors Activated by Adversarial Neuron Noise". In: *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. Ed. by Amir Globersons et al. 2024. URL: http://papers.nips.cc/paper%5C_files/paper/2024/hash/cfaccbd9b5e62562779351ebcb140c94-Abstract-Conference.html.

[177] Xiaoyun Xu et al. "Towards Backdoor Stealthiness in Model Parameter Space". In: *arXiv preprint arXiv:2501.05928* (2025).

[178] Xiong Xu et al. "Towards Reliable and Efficient Backdoor Trigger Inversion via Decoupling Benign Features". In: *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=Tw9wemV6cb.

[179] Yixiao Xu et al. "LT-Defense: Searching-free Backdoor Defense via Exploiting the Long-tailed Effect". In: *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. Ed. by Amir Globersons et al. 2024. URL: http://papers.nips.cc/paper%5C_files/paper/2024/hash/064f6bcd7d3c72fb187bfca35ba2bfd4-Abstract-Conference.html.

[180] Gang Yan et al. "DeFL: Defending against Model Poisoning Attacks in Federated Learning via Critical Learning Periods Awareness". In: *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*. Ed. by Brian Williams, Yiling Chen, and Jennifer Neville. AAAI Press, 2023, pp. 10711–10719. DOI: 10.1609/AAAI.V37I9.26271. URL: https://doi.org/10.1609/aaai.v37i9.26271.

[181] Lu Yan et al. “ParaFuzz: An Interpretability-Driven Technique for Detecting Poisoned Samples in NLP”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023. URL: http://papers.nips.cc/paper%5C_files/paper/2023/hash/d2b752ed4726286a4b488ae16e091d64-Abstract-Conference.html.

[182] Jinluan Yang et al. “Mitigating the Backdoor Effect for Multi-Task Model Merging via Safety-Aware Subspace”. In: *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL: https://openreview.net/forum?id=dqMqAaw7Sq.

[183] Wenhan Yang, Jingdong Gao, and Baharan Mirzasoleiman. “Better Safe than Sorry: Pre-training CLIP against Targeted Data Poisoning and Backdoor Attacks”. In: *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=ycLHJuLYuD.

[184] Wenhan Yang, Jingdong Gao, and Baharan Mirzasoleiman. “Robust Contrastive Language-Image Pretraining against Data Poisoning and Backdoor Attacks”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023. URL: http://papers.nips.cc/paper%5C_files/paper/2023/hash/2232e8fee69b150005ac420bfa83d705-Abstract-Conference.html.

[185] Yanxin Yang et al. “FilterFL: Knowledge Filtering-based Data-Free Backdoor Defense for Federated Learning”. In: *Proceedings of the 2025 on ACM SIGSAC Conference on Computer and Communications Security*. CCS ’25. Taipei, Taiwan, 2025.

[186] Yanxin Yang et al. “SampDetox: Black-box Backdoor Defense via Perturbation-based Sample Detoxification”. In: *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. Ed. by Amir Globersons et al. 2024. URL: http://papers.nips.cc/paper%5C_files/paper/2024/hash/dbb5180957513805ebeea787b8c66ac9-Abstract-Conference.html.

[187] Yuxin Yang et al. “Distributed backdoor attacks on federated graph learning and certified defenses”. In: *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 2024, pp. 2829–2843.

[188] Biao Yi et al. “Probe before You Talk: Towards Black-box Defense against Backdoor Unalignment for Large Language Models”. In: *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL: https://openreview.net/forum?id=EbxYDBhE3S.

[189] Jia-Li Yin et al. “Adversarial-Inspired Backdoor Defense via Bridging Backdoor and Adversarial Attacks”. In: *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*. Ed. by Toby Walsh, Julie Shah, and Zico Kolter. AAAI Press, 2025, pp. 9508–9516. DOI: 10.1609/AAAI.V39I9.33030. URL: https://doi.org/10.1609/aaai.v39i9.33030.

[190] Hao Yu et al. "DShield: Defending against Backdoor Attacks on Graph Neural Networks via Discrepancy Learning". In: *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society, 2025. URL: https://www.ndss-symposium.org/ndss-paper/dshield-defending-against-backdoor-attacks-on-graph-neural-networks-via-discrepancy-learning/.

[191] Yang Yu et al. "Untargeted Attack against Federated Recommendation Systems via Poisonous Item Embeddings and the Defense". In: *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*. Ed. by Brian Williams, Yiling Chen, and Jennifer Neville. AAAI Press, 2023, pp. 4854–4863. DOI: 10.1609/AAAI.V37I4.25611. URL: https://doi.org/10.1609/aaai.v37i4.25611.

[192] Danni Yuan et al. "Activation Gradient based Poisoned Sample Detection Against Backdoor Attacks". In: *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL: https://openreview.net/forum?id=VNMJfBBUd5.

[193] Zhuowen Yuan et al. "SHINE: Shielding Backdoors in Deep Reinforcement Learning". In: *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=nMWxLnSBGW.

[194] Rui Zeng et al. "CLIBE: Detecting Dynamic Backdoors in Transformer-based NLP Models". In: *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society, 2025. URL: https://www.ndss-symposium.org/ndss-paper/clibe-detecting-dynamic-backdoors-in-transformer-based-nlp-models/.

[195] Yi Zeng et al. "Adversarial Unlearning of Backdoors via Implicit Hypergradient". In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: https://openreview.net/forum?id=MeeQkFYVbzW.

[196] Yi Zeng et al. "Meta-Sift: How to Sift Out a Clean Subset in the Presence of Data Poisoning?" In: *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*. Ed. by Joseph A. Calandrino and Carmela Troncoso. USENIX Association, 2023, pp. 1667–1684. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/zeng.

[197] Yi Zeng et al. "Narcissus: A practical clean-label backdoor attack with limited information". In: *CCS*. 2023.

[198] Hangtao Zhang et al. "Test-Time Backdoor Detection for Object Detection Models". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*. Computer Vision Foundation / IEEE, 2025, pp. 24377–24386. DOI: 10.1109/CVPR52734.2025.02270. URL: https://openaccess.thecvf.com/content/CVPR2025/html/Zhang%5C_Test-Time%5C_Backdoor

%5C_Detection%5C_for%5C_Object%5C_Detection%5C_Models%5C_CVPR%5C_2025%5C_paper.html.

[199] Hanlei Zhang et al. "BARBIE: Robust Backdoor Detection Based on Latent Separability". In: *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society, 2025. URL: https://www.ndss-symposium.org/ndss-paper/barbie-robust-backdoor-detection-based-on-latent-separability/.

[200] Kaiyuan Zhang et al. "FLIP: A Provable Defense Framework for Backdoor Mitigation in Federated Learning". In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: https://openreview.net/forum?id=Xo2E217%5C_M4n.

[201] Quan Zhang et al. "AdvDoor: adversarial backdoor attack of deep learning system". In: *ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*. 2021.

[202] Yechao Zhang et al. "Secure Transfer Learning: Training Clean Model Against Backdoor in Pre-Trained Encoder and Downstream Dataset". In: *IEEE Symposium on Security and Privacy, SP 2025, San Francisco, CA, USA, May 12-15, 2025*. Ed. by Marina Blanton, William Enck, and Cristina Nita-Rotaru. IEEE, 2025, pp. 1–19. DOI: 10.1109/SP61157.2025.00162. URL: https://doi.org/10.1109/SP61157.2025.00162.

[203] Zaixi Zhang et al. "Backdoor defense via deconfounded representation learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 12228–12238.

[204] Zhiwei Zhang et al. "Robustness Inspired Graph Backdoor Defense". In: *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL: https://openreview.net/forum?id=trKNi4IUiP.

[205] Zhiyuan Zhang et al. "Fed-FA: Theoretically Modeling Client Data Divergence for Federated Language Backdoor Defense". In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023. URL: http://papers.nips.cc/paper%5C_files/paper/2023/hash/c39578c86423df5f9e8834ce1cd456e4-Abstract-Conference.html.

[206] Qi Zhao and Christian Wressnegger. "Two Sides of the Same Coin: Learning the Backdoor to Remove the Backdoor". In: *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*. Ed. by Toby Walsh, Julie Shah, and Zico Kolter. AAAI Press, 2025, pp. 22804–22812. DOI: 10.1609/AAAI.V39I21.34441. URL: https://doi.org/10.1609/aaai.v39i21.34441.

[207] Xingyi Zhao, Depeng Xu, and Shuhan Yuan. "Defense against Backdoor Attack on Pre-trained Language Models via Head Pruning and Attention Normalization". In: *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=1SiEfsCecd.

[208] Yue Zhao, Congyi Li, and Kai Chen. "UMA: Facilitating Backdoor Scanning via Unlearning-Based Model Ablation". In: *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*. Ed. by Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan. AAAI Press, 2024, pp. 21823–21831. DOI: 10.1609/AAAI.V38I19.30183. URL: https://doi.org/10.1609/aaai.v38i19.30183.

[209] Runkai Zheng et al. "Pre-activation Distributions Expose Backdoor Neurons". In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo et al. 2022. URL: http://papers.nips.cc/paper%5C_files/paper/2022/hash/76917808731dae9e6d62c2a7a6afb542-Abstract-Conference.html.

[210] Songzhu Zheng et al. "Topological detection of trojaned neural networks". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 17258–17272.

[211] Jiachen Zhou et al. "DataElixir: Purifying Poisoned Dataset to Mitigate Backdoor Attacks via Diffusion Models". In: *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*. Ed. by Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan. AAAI Press, 2024, pp. 21850–21858. DOI: 10.1609/AAAI.V38I19.30186. URL: https://doi.org/10.1609/aaai.v38i19.30186.

[212] Biru Zhu et al. "Moderate-fitting as a Natural Backdoor Defender for Pre-trained Language Models". In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo et al. 2022. URL: http://papers.nips.cc/paper%5C_files/paper/2022/hash/0799492e7be38b66d10ead5e8809616d-Abstract-Conference.html.

[213] Chaoyi Zhu, Stefanie Roos, and Lydia Y. Chen. "LeadFL: Client Self-Defense against Model Poisoning in Federated Learning". In: *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 43158–43180. URL: https://proceedings.mlr.press/v202/zhu23j.html.

[214] Liuwan Zhu et al. "SEER: Backdoor Detection for Vision-Language Models through Searching Target Text and Image Trigger Jointly". In: *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*. Ed. by Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan. AAAI Press, 2024, pp. 7766–7774. DOI: 10.1609/AAAI.V38I7.28611. URL: https://doi.org/10.1609/aaai.v38i7.28611.

[215] Mingli Zhu et al. “Neural Polarizer: A Lightweight and Effective Backdoor Defense via Purifying Poisoned Features”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023. URL: http://papers.nips.cc/paper%5C_files/paper/2023/hash/03df5246cc78af497940338dd3eacbaa-Abstract-Conference.html.

[216] Rui Zhu et al. “Selective Amnesia: On Efficient, High-Fidelity and Blind Suppression of Backdoor Effects in Trojaned Machine Learning Models”. In: *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*. IEEE, 2023, pp. 1–19. DOI: 10.1109/SP46215.2023.10351028. URL: https://doi.org/10.1109/SP46215.2023.10351028.

V

Final Insights

13

DISCUSSION

This concluding chapter explains the scientific and technical contributions of this thesis, as well as its limitations and promising future directions.

13.1. AUDIO BACKDOORS

Here, we discuss our contributions regarding the first research question.

> **RQ1 (Audio backdoors):** What types of audio triggers enable stealthy and robust backdoor attacks against automatic speech recognition under realistic assumptions?

In Chapter 3, we studied keyword spotting as a widely deployed speech recognition primitive and demonstrated that an adversary can insert a backdoor using an inaudible trigger. Assuming a data-poisoning threat model in which the attacker can modify only a small fraction of the training set, we show that poisoning just 1% of the data is sufficient to achieve a high attack success rate when the ultrasonic trigger is inserted into the input audio. This indicates that the model learns a shortcut feature that associates the trigger with the target label. We have explored the effect of various trigger properties on the attack success rate. In particular, among the parameters we varied, poisoning rate and trigger duration had the largest effect on the attack success rate. Additionally, we have created an Android keyword-spotting application that can be tricked by an inaudible signal played over the air and can generate an attacker-chosen misclassification. As this trigger is inaudible, adversaries could play it through their equipment the whole time that a user speaks to increase the chances of success. This is the first time a speech recognition backdoor has been verified in the physical world. An important observation from this work is that the attention layer from the LSTM architecture used in the experiments was more robust against our attack than the CNNs, resulting in a lower attack success rate.

In Chapter 4, we propose a backdoor attack that uses stylistic audio triggers implemented as signal transformations inspired by electric-guitar effects such as pitch shifting, distortion, and reverberation. Such transformations can be perceptually subtle and may

appear plausible in real environments. For example, reverberation can naturally arise from room acoustics. The trigger in this work is dynamic, as it depends on the input signal. Thus, the backdoored model could learn style features and associate them with the target label. This increases the attacker's flexibility in constructing realistic triggers that are compatible with a wide range of audio content. Our results indicate that even small, structured changes in audio "style" can be sufficient to induce distinct model behaviors.

In Chapter 5, we propose a backdoor attack that uses emotional prosody as an audio trigger to insert a backdoor in speaker identification. We verified the functionality across multiple experimental setups and demonstrated that it can bypass various preprocessing defenses, such as quantization, median filtering, and squeezing. However, our attack did not always bypass a pruning-based defense, as in some cases the backdoor-related neurons could be pruned by the defense, decreasing the attack success rate significantly (up to 41.4% decrease). Experimental evidence indicated that pruning a few neurons across multiple layers is more effective than pruning multiple neurons in a single layer.

The central message from this part of the thesis is that keyword spotting and speaker identification applications can be backdoored. However, each approach has distinct characteristics and trade-offs we need to consider. A trigger that is a tone at a specific frequency (e.g., a 21 kHz ultrasonic tone) can be easily learned from models but can also be mitigated by an adaptive adversary through simple signal-processing defenses such as filtering. Dynamic triggers are harder to create, but they also become stealthier because simple countermeasures, such as filtering, may not be able to remove them from poisoned samples. We have also explored different styles and emotions as triggers that resulted in slight variations in the attack's performance. Such variations show that each trigger can lead to different features learned by our models in different ways. Thus, the attacker could optimize our approaches if the aim is to create the most effective attack. A positive side effect of our triggers is that since they introduce controlled, interpretable changes in a signal, we can use them as explainability tools to understand how and what a neural network learns during training.

13.2. NON-VISION BACKDOORS

RQ2 (Non-vision backdoors): How can backdoor attacks be instantiated across non-vision modalities like graphs (GNNs), tabular data, and text, so that they remain effective, utility-preserving, and hard to detect under realistic pipelines and defenses?

In Chapter 6, we proposed a backdoor in Federated GNNs using attacker-designed subgraphs as triggers. We evaluated two approaches for this attack. First, we defined a distributed backdoor attack (DBA). In DBA, we assume multiple (i.e., at least two) malicious clients that hold their own local trigger, which is inserted into their local dataset. All malicious clients share a common malicious goal (e.g., misclassification to a specific target class). Second, we defined a centralized backdoor attack (CBA). In CBA, we assume a single malicious client that uses a global trigger, a superposition of all the local triggers used in our DBA scenario. This global trigger is again used to poison the local dataset of this malicious client. In most cases, DBA achieves a higher attack success rate than CBA. We also showed that increasing the number of malicious clients in DBA leads to

an increased attack success rate. An important observation is that while in CBA we train with a global trigger (superposition of all the local triggers used in the DBA scenario), at test time any single local trigger still activates the backdoor. Finally, we showed that our attacks could bypass FoolsGold, a defense made to spot malicious updates from poisoned clients in Federated Learning.

In Chapter 7, we introduce CatBack, a backdoor attack for tabular data that assumes an adversary with access to the training pipeline. CatBack computes a universal trigger as an optimized perturbation using gradient descent. To handle mixed feature types (categorical and numerical), we first map categorical features to floating-point values using our novel frequency-based encoding, which enables end-to-end optimization in a unified continuous space. We then optimize an objective that promotes the attacker-chosen target label and high-frequency feature values of the data distribution while constraining the perturbation magnitude, yielding a universal perturbation that can be added to poisoned samples. In the black-box setting, we construct the perturbation on a surrogate open-source model. We evaluate CatBack on five datasets and four models, including neural networks and classical machine learning methods, and obtain consistently high attack success rates. Across most experimental settings, CatBack also bypasses the defenses we tested (Spectral Signatures, Neural Cleanse, Beatrix, and Fine-Pruning).

In Chapter 8, we perform a systematic evaluation of backdoor attacks across multiple domains, aiming to identify common patterns and modality-specific differences in backdoor behavior. Specifically, we implement both dirty-label and clean-label attacks and evaluate them in four application settings: image classification, text classification, audio classification, and graph learning. Across these domains, we study how trigger characteristics, such as poisoning rate, trigger size, and trigger placement, affect both the attack success rate and the clean accuracy drop. Our results show that backdoor injection is generally feasible across all tested modalities without noticeably degrading clean accuracy. Poisoning rate consistently ranks among the most influential factors in our attack. We also confirm that clean-label backdoors are often substantially less effective than dirty-label attacks under realistic poisoning budgets. However, a strong adversary could generate the clean-label triggers through optimization to improve the attack's performance. Finally, we note that while some trends (e.g., larger triggers often improving attack success) transfer across domains, other properties, such as the impact of trigger position, depend heavily on the model architecture and data modality, underscoring the need for careful, domain-aware evaluation of backdoor threats.

In Chapter 9, we present Backdoor Pony, a web-based, user-friendly framework for evaluating backdoor attacks and defenses across multiple application domains. Backdoor Pony addresses a key limitation of prior toolboxes, which largely focus on computer vision and often require complex command-line pipelines, by providing an interactive GUI-driven workflow that supports image, text, audio, and graph tasks within a unified interface. The framework follows a modular client–server architecture (front-end GUI + back-end execution engine) and is packaged with Docker for reproducibility and GPU support, allowing users to configure datasets, models, attack/defense hyperparameters, and automatically generate standardized metrics and plots such as attack success rate and clean accuracy drop. Importantly, Backdoor Pony is designed to be extensible, enabling researchers to integrate new attacks and defenses through an extension API, which helps

keep pace with the rapidly evolving backdoor literature. Overall, this tool lowers the barrier to entry for backdoor research, facilitates systematic ablation studies, and provides a practical foundation for more consistent and comparable evaluations across domains.

The central conclusion of this part of the thesis is that backdoor attacks are not a computer-vision–specific phenomenon, but a general phenomenon that extends to diverse non-vision modalities and learning pipelines. Across federated GNNs, tabular data, and language and audio tasks, we show that effective backdoors can be instantiated through appropriate triggers (e.g., attacker-designed subgraphs in GNNs or optimized universal perturbations in tabular data) while largely preserving benign utility. The core idea of the data poisoning scenario remains the same, but the techniques may vary by data modality. For this reason, defenses targeting computer vision backdoors may be bypassed, as different techniques are required across modalities. This highlights the need to create modality-agnostic countermeasures that can detect backdoors without relying on the application domain. Finally, by releasing an evaluation framework that unifies experiments across domains, this part of the thesis provides practical tools for reproducible and easy evaluation of backdoor attacks.

13.3. COMPREHENSIVE STEALTHINESS

RQ3 (Comprehensive stealthiness): To what extent can backdoor attacks achieve stealthiness against heterogeneous defenses that are based on input-, feature-, and parameter-space analyses?

In Chapter 10, we propose BAN, a defense against backdoors that assumes white-box access to the model and its parameters. Previous work has shown that backdoored models tend to behave differently from clean models in the presence of noise. BAN exploits this finding. In particular, BAN finds an adversarial perturbation that maximizes the model's loss on clean inputs by solving a maximization problem with the PGD algorithm. In backdoored models, such noise could amplify the backdoor behavior. Thus, when misclassifications are mostly toward a specific class, we flag the model as suspicious and consider that class as the backdoor's target class. To achieve consistent results, we have also used a trainable mask that decouples the model's clean and backdoored features. Previous works in this area also used a trainable mask, but without a size regularizer (L1). In such cases, the optimization could converge to degenerate solutions and non-informative masks (e.g., a mask full of 1s). Based on this detection mechanism, we proposed a backdoor removal technique that fine-tunes the model to minimize classification loss under the learned perturbation, reducing the backdoor behavior. In our experiments, BAN outperforms other baselines consistently across multiple datasets, architectures, and representative attacks, without affecting clean accuracy. BAN is also faster. An interesting observation is that state-of-the-art defenses may be effective against modern attacks like WaNet but not against older attacks like BadNets. Our experiments indicated that while WaNet is stealthy in input space, it is strong in feature space. However, BadNets is visually obvious, but its feature footprint can be weaker. BAN, using adversarial noise, amplifies the backdoor effect, making even weak backdoor triggers in the feature space distinguishable.

In Chapter 11, we investigate backdoor stealthiness beyond the input and feature

space and highlight model parameter space as a critical, yet often overlooked, axis of robustness. We first conduct a large-scale evaluation of 12 representative backdoor attacks against 17 diverse defenses, and show that many state-of-the-art attacks that aim to be stealthy in the input or feature space remain vulnerable to parameter-space defenses, revealing a key blind spot in current evaluation practices. To explain this behavior, we analyze backdoor-related neurons in parameter space and observe that many attacks introduce prominent parameter anomalies that defenders can exploit. Motivated by these findings, we propose Grond, a novel supply-chain backdoor attack that targets comprehensive stealthiness, i.e., being stealthy simultaneously in input, feature, and parameter space. Grond uses an imperceptible universal PGD (UPGD) trigger for input-space stealthiness and introduces an Adversarial Backdoor Injection (ABI) module that adaptively suppresses prominent backdoor-related neurons during training (guided by channel-wise sensitivity measures), thereby improving parameter-space stealthiness. Extensive experiments on CIFAR10, GTSRB, and ImageNet200 show that Grond consistently achieves high attack success rates while bypassing a broad range of pruning-, fine-tuning-, and detection-based defenses. We further demonstrate that ABI can strengthen the robustness of several existing backdoor attacks against parameter-space defenses.

From the above works, we see that different backdoor triggers yield different internal representations in backdoored models. Additionally, backdoor countermeasures could target one or more types of these representations. These observations can help us understand what the models learn and how they map a trigger internally, which could lead to better defenses in the future. We also show that stealthy backdoors in all three aspects (input-, feature-, and parameter-space) are possible if the adversary has white-box access to the model's weights.

13.4. STANDARDIZED DEFENSE EVALUATION

> **RQ4 (Standardized defense evaluation):** What is the current state in evaluating backdoor defenses, and how can we define a reproducible and comprehensive standard that enables fair comparison across threat models, data modalities, and deployment settings?

In Chapter 12, we present a systematic study of all the backdoor defenses that have been published in top cybersecurity (i.e., Usenix Security, S&P, CCS, and NDSS) and AI conferences (i.e., ICLR, ICML, NeurIPS, CVPR, and AAAI). We have analyzed 183 papers and created a dataset with various characteristics for each paper. Our analysis resulted in various insights. For example, we pointed out that in many cases, a defense is not evaluated against diverse experimental settings. As a result, we cannot understand its capabilities in depth and point out its strengths and weaknesses. Additionally, instead of evaluating a defense against many attacks of the same type, it is better to show its performance against attacks that use triggers with different levels of stealth. Another important observation is that defense overhead is not usually reported. Also, the defense's effect on the model's performance if there is no backdoor is mostly omitted. To complement the survey, we also performed a large-scale empirical re-evaluation of representative defenses across multiple defense families (pre-training, in-training, and post-training), against five

widely used attacks, on datasets spanning increasing difficulty (from MNIST to CIFAR-100 and ImageNet-1K) and across various model architectures, including CNNs and a Vision Transformer. Our results demonstrate that defense effectiveness can vary substantially across evaluation setups and often does not generalize from small benchmarks to realistic settings. This shows the importance of a common framework that everyone uses to understand a defense's capabilities and fairly compare a new defense with the state-of-the-art. Finally, we provide actionable recommendations towards a more standardized evaluation of backdoor defenses. Our datasets that summarize the literature analysis are open-sourced to support standardized and transparent future evaluation.

13.5. LIMITATIONS

While the thesis covers diverse modalities and learning pipelines, several limitations should be highlighted to bound the conclusions.

- **No experiments with industrial speech models**: In the first part of the thesis, we discussed backdoors in keyword spotting and speaker identification. These applications are important but do not fully capture the breadth of modern deployments, such as streaming automatic speech recognition, large-vocabulary speech recognition, or multi-turn dialogue systems. Such models learn to recognize characters in speech under various conditions, making them more robust to simple signal properties, such as ultrasonic signals. Additionally, as they classify different phonemes, it may be more difficult to associate styles or emotions with a specific phrase. Such models are used in commercial applications like Siri, Alexa, or Google Assistant and are trained with a vast amount of data, which could not be easily poisoned. Thus, to demonstrate the impact of backdoors in speech recognition, we need to evaluate these models under realistic assumptions.

- **Limited model architectures and toy pipelines**: In a large part of our experiments, we used CNNs and simple research-oriented pipelines. While CNNs are used in a wide variety of applications, newer models are now taking over. It is important to investigate whether our insights can generalize to these new models. Additionally, AI pipelines in the industry often include preprocessing, postprocessing, and confidence-based rejection rules that we did not model in our setups. Thus, the observed behaviors should be interpreted as strong evidence of feasibility, not as an exhaustive characterization of all deployments in practice.

- **Lack of explainability tools in results interpretation**: In our experiments, there are various cases where the explanation of the behavior is not straightforward, so we wrote our intuition about them. However, a better approach would be to use explainability tools to find the real causes of the observed behaviors.

- **Usage of common datasets**: In most experiments, we have used publicly available datasets that are mostly used in research papers. While common datasets enable comparability, they may under-approximate the complexity of data distributions found in deployed systems. For example, computer vision datasets such as ImageNet may have different properties from malware datasets. Because attackers can

exploit data in different ways, it is crucial to investigate the backdoor's behavior across different datasets (and types of data).

- **No formal proofs**: Our works are mostly empirical and based on empirical evaluation and results. Our insights are not based on formal studies or formal mathematical proofs, which may lead to less generalizable results.

13.6. DISCUSSION & FUTURE WORK

In this thesis, we demonstrated the feasibility of backdoors under various assumptions and in different modalities. However, our investigation is not exhaustive, and there are multiple directions for future research.

13.6.1. REAL-WORLD BACKDOOR FEASIBILITY

To assess the real-world impact of backdoor threats, future work should prioritize realistic experimental setups and physical-world validation. In practice, a backdoor attacker must not only compromise training (e.g., via data poisoning or supply-chain threat model) but also possess a feasible trigger-delivery channel at inference time. For many optimization-based computer-vision triggers, this second requirement may be the limiting factor: imperceptible digital perturbations can be difficult to realize under camera capture, viewpoint changes, and environmental variability. This suggests that physically realizable triggers, such as simple, robust visual artifacts, may be more relevant to deployed systems than highly optimized but fragile patterns.

Analogously, extending physical-world evaluation to speech is essential. Models used in automatic speech recognition differ substantially from those used in keyword spotting in architecture and inference dynamics (e.g., sequence decoding and language constraints), which may affect the effectiveness of ultrasound-, style-, or emotional prosody-based triggers. Therefore, a key next step is to evaluate whether these triggers remain effective and stealthy under realistic recording/playback channels and against modern ASR pipelines.

13.6.2. SECURITY CRITICAL DOMAINS

Future work should expand the study of backdoors beyond standard image classification benchmarks to security-critical ML applications such as malware detection, intrusion detection, and spam filtering. These domains are directly relevant to adversarial settings and often operate under different data characteristics (e.g., sparse and discrete features, strong structural constraints, distribution shift, unbalanced data), which may influence both the feasibility of trigger injection and the mechanisms by which backdoors are inserted into the models. Although the backdoor ultimately is encoded in the model's parameters, comparing backdoor behavior across heterogeneous security datasets can yield more general insights into backdoor learning dynamics. Moreover, these application settings may enable domain-aware defenses that exploit semantic or structural validity constraints (e.g., executable-format consistency or protocol-level invariants) to detect or suppress backdoor activation. In addition, these domains often provide richer telemetry and post-deployment monitoring, enabling detection strategies that are not available in static image classification benchmarks.

13.6.3. BRIDGING THE GAP BETWEEN RESEARCH AND INDUSTRY

As demonstrated throughout this thesis, backdoors can be injected into trained models under common training pipelines and small poisoning rates. However, there are no publicly documented cases of AI backdoors in production ML systems[1], which may reduce the perceived urgency for deploying backdoor defenses in industry. Recent work argues that this gap between demonstrated feasibility and industrial adoption is partly driven by the engineering cost of integrating backdoor defenses, and proposes pipeline-compatible approaches that audit and improve backdoor resistance with minimal disruption to existing training workflows (e.g., via configuration search) [1]. Motivated by this, an important direction is to design countermeasures that impose low overhead and require minimal changes to end-to-end ML pipelines, thereby lowering the adoption barrier and enabling systematic, real-world feedback about backdoors. In parallel, since many backdoor threats originate from data poisoning and supply-chain uncertainty, strengthening data governance is a promising complementary direction, including dataset/model versioning, data deduplication, anomaly detection, and auditable lineage for training data and model checkpoints. Finally, training-time robustness mechanisms (e.g., regularization and augmentation strategies that reduce shortcut learning from small subsets of data) may increase the poisoning budget required for successful backdoor insertion while preserving benign utility.

13.6.4. STANDARDIZED AND REPRODUCIBLE EVALUATION

The backdoor literature is rapidly evolving, with frequent new attacks and defenses. In this context, periodic and systematic re-evaluation of existing methods is valuable because reported improvements are often sensitive to experimental choices and may not generalize across threat models, modalities, or architectures. Incentivizing replication and standardized re-evaluation would clarify the strengths and weaknesses of each method and enable fair comparison under explicit assumptions, rather than relying on isolated benchmark results. This thesis further demonstrates that both attacker-side and defender-side assumptions substantially shape empirical outcomes, reinforcing the need for explicit and standardized threat-model reporting. A practical direction is to expand unified tool support for backdoor research across modalities and defense families. Such tooling can enforce consistent metrics (e.g., attack success rate, clean accuracy drop, computational overhead, and clean-model side effects), reduce hidden degrees of freedom in evaluation, and make ablations routine rather than exceptional. Motivated by our findings in Chapter 11, future evaluation standards should explicitly test defenses against attacks with diverse stealthiness profiles and artifacts in input-, feature-, and parameter-space, ideally within a unified benchmark suite. Concretely, the community could define a minimum evaluation protocol that includes (i) multiple attack families (e.g., input-space, feature-space, and parameter-stealthy attacks), (ii) multiple poisoning rates, and (iii) multiple configurations (dirty-label vs. clean-label; source-specific vs. source-agnostic), (iv) multiple datasets/architectures, (v) overhead reporting (time/memory), and (vi) adaptive attacker/defender. These guidelines could be implemented via open-source frameworks (e.g., BackdoorBench [4]) to improve reproducibility and comparability, and potentially

[1] Researchers have found models with malware in Hugging Face (https://atlas.mitre.org/studies/AML.CS0031), however, here we mean AI backdoors that compromise the model's decisions.

extended into a community-maintained benchmark with appropriate safeguards against overfitting (e.g., periodic refresh and transparent reporting of overhead and failure cases).

13.6.5. BACKDOOR THREATS IN FOUNDATION MODELS

Multimodal foundation models have recently emerged and are increasingly adapted to downstream tasks via low-data fine-tuning. Their scale and multimodal nature may expand the backdoor attack surface, enabling triggers that exploit cross-modal interactions (e.g., image and text combinations) rather than perturbations confined to a single input channel. Therefore, future work should investigate backdoor threats and defenses in multimodal settings, with emphasis on modality-agnostic countermeasures that rely on invariants not tied to a particular application or input representation (e.g., consistent anomalies in model behavior, representations, or parameter-space anomalies).

As foundation models continue to improve and platforms such as Hugging Face enable widespread distribution of pretrained checkpoints, future work should evaluate backdoor threats under a fine-tuning-based threat model. In this setting, an attacker (e.g., a malicious model publisher or a compromised repository) inserts a backdoor into a pretrained model that is later adapted to a downstream task via fine-tuning. This raises practical questions about whether backdoors persist, attenuate, or transform during transfer to new tasks and data distributions. Accordingly, a promising defense direction is to move beyond "train-from-scratch" assumptions and develop methods that operate on pretrained models with white-box access, detecting or mitigating backdoors before or during fine-tuning with minimal disruption to standard adaptation pipelines.

13.6.6. BACKDOORS, OBSERVABILITY, AND INTERPRETABILITY

Although defending against backdoors remains challenging, especially under adaptive attackers and supply-chain threat models, improving the observability of deployed ML systems is important. Instrumentation and auditing artifacts (e.g., prediction confidence, intermediate representations, and feature-importance information) can support forensic analysis and debugging when anomalous behavior is observed, including potential trigger-activated misclassifications.

This motivates further research at the intersection of backdoor security and model interpretability. Interpretability methods may not directly guarantee robustness, but they can provide actionable signals for monitoring, root-cause analysis, and defense design. Moreover, backdoor triggers can serve as an experimental probe for understanding learning dynamics: they enable tests of whether models rely on sparse shortcut features, how such features are encoded in internal representations, and how effectively they can be removed or "unlearned" through fine-tuning, pruning, or other mitigation procedures.

13.6.7. CONNECTING BACKDOORS TO LLM JAILBREAKING

Large language models (LLMs) have attracted substantial attention in academia and industry, particularly following the introduction of conversational systems such as ChatGPT [3]. These models are pretrained on large-scale web corpora and subsequently aligned using additional objectives and safety policies. Nevertheless, it has been shown that carefully crafted prompts can sometimes elicit behaviors that violate intended safety constraints, a phenomenon often referred to as jailbreaking.

Although jailbreaking and backdoor attacks arise from different threat models, they share a structural similarity: a specific input pattern (a prompt or a trigger) can induce a targeted and undesirable model behavior while the model remains broadly useful on benign inputs. This suggests an interesting research direction at the interface of backdoor robustness and LLM safety: to investigate whether concepts and countermeasures from the backdoor literature (e.g., anomaly-based detection, robustness auditing, and mitigation via targeted fine-tuning) can be adapted to improve LLM resistance to malicious prompting, and to define standardized metrics that jointly capture safety, utility, and attacker adaptivity in this setting.

13.7. CONCLUSIONS

Similar to classical cybersecurity, where large-scale deployment and internet connectivity revealed new attack surfaces that motivated defense-in-depth, widespread ML deployment introduced adversarial machine learning, including backdoor attacks. Backdoors in AI are challenging to eliminate and difficult to certify the absence of in a given model. As discussed in [2], the backdoor threat model gives the adversary many degrees of freedom. For example, under a supply-chain threat model, an attacker can train a model with full control over the training pipeline (dataset, optimization procedure, hyperparameters, and weights) and distribute it via public repositories (e.g., Hugging Face) for downstream users to fine-tune or deploy. In this setting, the attacker may manipulate training data, design optimized perturbations that create specific internal representations for selected inputs, or even directly modify model parameters. Moreover, obtaining formal guarantees about model behavior is computationally intractable in general [2]. For these reasons, we argue that universal countermeasures that defend against arbitrary backdoor attacks are unlikely. Nevertheless, this should not discourage research: systematic study of attacks and defenses can yield actionable insights, improve robustness in practice, and inform guidelines and engineering practices that raise the cost of inserting backdoors in real systems.

REFERENCES

[1] Eugene Bagdasarian and Vitaly Shmatikov. “Mithridates: Auditing and boosting backdoor resistance of machine learning pipelines”. In: *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 2024, pp. 4480–4494.

[2] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. “Handcrafted backdoors in deep neural networks”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 8068–8080.

[3] OpenAI. *Introducing ChatGPT*. Nov. 30, 2022. URL: https://openai.com/blog/chatgpt (visited on 12/30/2025).

[4] Baoyuan Wu et al. “BackdoorBench: A Comprehensive Benchmark of Backdoor Learning”. In: 2022.

CURRICULUM VITÆ

Stefanos KOFFAS

Stefanos Koffas was born in Athens, Greece. He received his Master of Engineering (MEng.) in Electrical and Computer Engineering from the National Technical University of Athens, Greece, in May 2016. After working for 2.5 years as a DevOps engineer and a system administrator in Greece, he started his Master of Science (MSc.) in Computer Engineering at Delft University of Technology in September 2018, where he did his master's thesis on backdoor attacks in deep learning, a fascinating and novel topic. As a result, Stefanos started his Ph.D. in July 2021, joining the cybersecurity research group at the Delft University of Technology, supervised by Professors Mauro Conti and Stjepan Picek. His main research interest is backdoor attacks in AI. During his Ph.D., he investigated how backdoors can be applied to different modalities of AI systems and how we can defend against them. He has presented his work at prestigious international conferences, reviewed several papers, gave invited talks, and also supervised multiple bachelor's and master's students.

LIST OF PUBLICATIONS

PUBLISHED

21. M. Arazzi, **S. Koffas**, A. Nocera, S. Picek (2024). Let's Focus: Focused Backdoor Attack against Federated Transfer Learning. *Neurocomputing, Volume 696, 2026.*

20. B. Tajalli, **S. Koffas**, S. Picek, (2026 February). CatBack: Universal Backdoor Attacks on Tabular Data via Categorical Encoding. *In Proceedings of the 33rd Annual Network and Distributed System Security Symposium (NDSS) (Vol. 2026).*

19. M. G. Collu, T. Janssen-Groesbeek, **S. Koffas**, M. Conti, S. Picek, (2026). Dr. Jekyll and Mr. Hyde: Two faces of LLMs. *Italian Conference on CyberSecurity (ITASEC) 2026.*

18. X. Xu, Z. Liu, **S. Koffas**, S. Picek. 2025. Towards Backdoor Stealthiness in Model Parameter Space. *In Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25).* Association for Computing Machinery, New York, NY, USA, 2863–2876.

17. B. Pleiter, B. Tajalli, **S. Koffas**, G. Abad, J. Xu, M. Larson, S. Picek. (2025, September) Long Paper: Backdoor Attacks on Transformers for Tabular Data: An Empirical Study. *In European Symposium on Research in Computer Security (ESORICS) International Workshops.* Cham: Springer Nature Switzerland.

16. J. te Lintelo, **S. Koffas**, S. Picek. (2025). The SkipSponge attack: Sponge weight poisoning of deep neural networks. *ITU Journal on Future and Evolving Technologies, 6(3), 247–263.* https://doi.org/10.52953/xkbu4341

15. S. Gaure, **S. Koffas**, S. Picek, S. Rønjom, (2025, April). $L^2 \cdot M = C^2$ Large Language Models Are Covert Channels. *In ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 1-5).*

14. X. Xu, Z. Liu, **S. Koffas**, S. Yu, S. Picek, (2024). BAN: detecting backdoors activated by adversarial neuron noise. *Advances in Neural Information Processing Systems, 37, 114348-114373.*

13. B. Tajalli, **S. Koffas**, G. Abad, S. Picek, (2024, November). ELMs Under Siege: A Study on Backdoor Attacks on Extreme Learning Machines. *In Proceedings of the 2024 Workshop on Artificial Intelligence and Security (pp. 125-136).*

12. C. Schoof, **S. Koffas**, M. Conti, S. Picek (2024, November). Emoback: Backdoor attacks against speaker identification using emotional prosody. *In Proceedings of the 2024 Workshop on Artificial Intelligence and Security (pp. 137-148).*

11. C. Schoof, **S. Koffas**, M. Conti, S. Picek (2024, May). Beyond phantomsponges: Enhancing sponge attack on object detection models. *In Proceedings of the 2024 ACM Workshop on Wireless Security and Machine Learning (pp. 14-19).*

10. C. Kreza, **S. Koffas**, B. Tajalli, M. Conti, S. Picek, (2024, May). Backdoors on Manifold Learning. *In Proceedings of the 2024 ACM Workshop on Wireless Security and Machine Learning (pp. 1-7)*.

9. H. Cai, P. Zhang, H. Dong, Y. Xiao, **S. Koffas**, Y. Li, Toward Stealthy Backdoor Attacks Against Speech Recognition via Elements of Sound in *IEEE Transactions on Information Forensics and Security, vol. 19, pp. 5852-5866, 2024*, https://doi.org/10.1109/TIFS.2024.3404885.

8. J. Xu, **S. Koffas**, S. Picek, (2024). Unveiling the threat: Investigating distributed and centralized backdoor attacks in federated graph neural networks. *Digital Threats: Research and Practice, 5(2), 1-29*.

7. **S. Koffas**, B. Tajalli, J. Xu, M. Conti, S. Picek (2023). A Systematic Evaluation of Backdoor Attacks in Various Domains. *In Embedded Machine Learning for Cyber-Physical, IoT, and Edge Computing: Use Cases and Emerging Challenges (pp. 519-552)*. Cham: Springer Nature Switzerland.

6. J. Xu, **S. Koffas**, O. Ersoy, S. Picek, (2023, July). Watermarking graph neural networks based on backdoor attacks. *In 2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)* (pp. 1179-1197).

5. **S. Koffas**, L. Pajola, S. Picek, M. Conti (2023, June). Going in style: Audio backdoors through stylistic transformations. *In ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1-5).

4. A. Mercier, N. Smolin, O. Sihlovec, **S. Koffas**, S. Picek, Backdoor Pony: Evaluating backdoor attacks and defenses in different domains, *SoftwareX, Volume 22, 2023, 101387, ISSN 2352-7110*.

3. J. Xu, R. Wang, **S. Koffas**, K. Liang, S. Picek (2022, December). More is better (mostly): On the backdoor attacks in federated graph neural networks. *In Proceedings of the 38th Annual Computer Security Applications Conference (pp. 684-698)*.

2. **S. Koffas**, J. Xu, M. Conti, S. Picek, (2022, May). Can you hear it? backdoor attacks via ultrasonic triggers. *In Proceedings of the 2022 ACM workshop on wireless security and machine learning* (pp. 57-62).

1. **S. Koffas**, S. Picek, M. Conti (2022, June). Dynamic backdoors with global average pooling. *In 2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)* (pp. 320-323).

UNDER SUBMISSION

4. G. Abad, M. Krček, **S. Koffas**, B. Tajalli, M. Arazzi, R. Riaño, X. Xu, Z. Liu, A. Nocera, and S. Picek. SoK: The Last Line of Defense: On Backdoor Defense Evaluation. arXiv preprint arXiv:2511.13143 (2025).

3. M. Arazzi, M. Conti, **S. Koffas**, M. Krcek, A. Nocera, S. Picek, J. Xu (2023). Label inference attacks against node-level vertical federated GNNs. arXiv preprint arXiv:2308.02465.

2. M. Conti, N. Farronato, **S. Koffas**, L. Pajola, S. Picek (2023). Invisible threats: backdoor attack in OCR systems. arXiv preprint arXiv:2310.08259.

1. G. Abad, J. Xu, **S. Koffas**, B. Tajalli, S. Picek, M. Conti. (2023). Sok: A systematic evaluation of backdoor trigger characteristics in image classification. arXiv preprint arXiv:2302.01740.

ACKNOWLEDGEMENTS

In this section I would like to express my gratitude for all the people that made my Ph.D. journey possible, enjoyable, and a life-changing experience.

First of all, I am deeply thankful to my daily supervisor, Stjepan Picek, for giving me this opportunity and believing in me. I am especially grateful for his guidance, patience, constant availability, and continuous support throughout my Ph.D., and for his insightful feedback and encouragement, which helped me grow both as a researcher and as a person. His support and guidance made this project possible. I am also grateful to my promotor, Mauro Conti, for his clear advice, the intellectual freedom he provided during this project, and the opportunities he created for me to expand my network through collaborations with great researchers. Last but not least, I would like to thank Georgios Smaragdakis for stepping in and becoming my co-promotor. I feel very lucky to have such a strong supervisory team to guide me through the ups and downs of my Ph.D. journey. Without their support, completing this dissertation would not have been possible.

Additionally, I would like to thank Megha Khosla for serving as a committee member for my Go/No-Go meeting. I am also grateful to the members of my Ph.D. defense committee for their time, careful reading of this thesis, and valuable feedback.

During these years, I have crossed paths and collaborated with many wonderful people who helped me (directly or indirectly) complete my Ph.D. First, I would like to thank my close collaborators for the many papers we wrote together. In particular, I want to thank Jing Xu, Gorka Abad, Behrad Tajalli, Marco Arazzi, and Luca Pajola for all the joyful moments, but also for the frustrations we shared over the years. Our bond grew strong through life-changing experiences: working side by side to meet deadlines, brainstorming ideas, crafting the best possible replies to reviewers, attending conferences and exploring different countries together, and even going out on King's Day. My Ph.D. journey was far more enjoyable because of you.

I would also like to express my gratitude to Zhuoran Liu, Xiaoyun Xu, Rui Wang, Kaitai Liang, Nicola Farronato, Oguzhan Ersoy, Hanbo Cai, Pengchecng Zhang, Hai Dong, Yan Xiao, Yiming Lim, Antonino Nocera, Roberto Riaño, Simen Gaure, Sondre Rønjom, and Martha Larsson for the papers we wrote together. While not every paper is included in this dissertation, your contributions were invaluable to the completion of my Ph.D.

I feel very privileged to have supervised students during my Ph.D. I learned a lot from each of you, and I am confident you will be successful in whatever path you choose. I am especially grateful for the opportunity to collaborate closely with many of you, including through co-authored papers, which contributed meaningfully to this dissertation. I would like to thank Massimiliano Belluomini, Alex Simonov, Julian van der Horst, Coen Schoof, Jona te Lintelo, Christina Kreza, Enrico Leka, Arthur Mercier, Oliver Sihlovec, Nikita Smolin, and Selena Mendez.

Additionally, I would like to express my gratitude to my LL Corsairs CTF teammates, Matteo Gioele Collu and Tom Janssen-Groesbeek. When we started working together, we

were still finding our way in the world of LLMs, but over time we learned a great deal and grew together, especially in understanding LLM security. Participating with you in LLM CTFs has been a real joy.

Additionally, I feel thankful for all the people we shared an office with even if it was for a brief time. Martin Mladenov, Yuqian Song, Maarten Weyns, Murtuza Mohammed, Dario Ferrero, Jesus Maria Gomez Moreno I really enjoyed our short coexistence in the corner office and the hoops in the little basket on the door (:P). Jing Xu and Huimin Li, I learned so much from you, and I always enjoyed our conversations about Chinese culture and history. Marina, you always had something fun to say and I suspect I would still lose our bet, as my browser tabs have only increased.

I owe many of my best times in the Netherlands to the TU Delft CYS group (and a few people from other groups), who also contributed positively to my Ph.D. journey. I express my gratitude to Alexios Voulimeneas, Giovanne Moura, Harm Griffioen, Roland Kromes, Lilika Markatou, Enrico Bassetti, Tjitske Koster, Azade Rezaeezade, Vissarion Moutafis, Dazhuang Liu, Azqa Nadeem, Jelle and Daniel Vos, Robert Baumgartner, Sicco, Marwan Khabbaz, Jorrit van Assen, Zeki Erkin, Andrea Agiollo, Enkelenda Bardhi, Andreas Athanasiou, Sandra Wolf, Ruud de Jong, Clinton Cao, Florine Dekker, Stefano Cecconello, Guilherme Perin, HuanHuan Chen, Luca Marriot, Leo Weissbart, Tianyu Li, Lichao Wu, Yanqi Qiao, Jinke He, Antreas Dionysiou.

I would also like to thank people from the TPM group (Savvas Zannettou, Yury Zhauniarovich, Marie-Therese Sekwenz, Simon Parkin, Szu-Chun Huang) for including me into their Thursday-night tradition.

Additionally, I would like to express my gratitude to colleagues from Radboud University who were not mentioned in the previous paragraphs. In particular, I would like to thank Yanis Belkheyar, Konstantina Miteloudi, and Lukasz Chielewski for our discussions and for the great time we shared in Croatia, as well as Zahra Moti Jeshveghani for attending CCS '25 together.

Also, I am very grateful to my colleagues at the University of Padua and the SPRITZ group. In particular, I would like to thank Francesco Marchiori, Tommaso Bianchi, Denis Donadel, Ying Yuan, Federico Turrin, Pier Paolo Tricomi, and Jiaxin Li for the great discussions, the time we shared, and the bacaro tour in Venice. A special shoutout goes to Alberto Castagnaro for the unforgettable moments we shared at two CCS conferences.

I would also like to thank my friends Fiske, Chris, Ravi, and Alex for bringing fun and balance to my time in the Netherlands. Special thanks to Fiske for checking the AI-generated Dutch summary of my thesis. I am also grateful to have been a member of an amazing amateur football team in the Hague (SC Remo), which helped me stay happy and positive.

Finally, I want to thank my parents, my brother, my childhood friends in Greece, and my partner Theodora who are always there for me and support me unconditionally. Finishing the Ph.D. would not have been possible without you in my life.


TUDelft