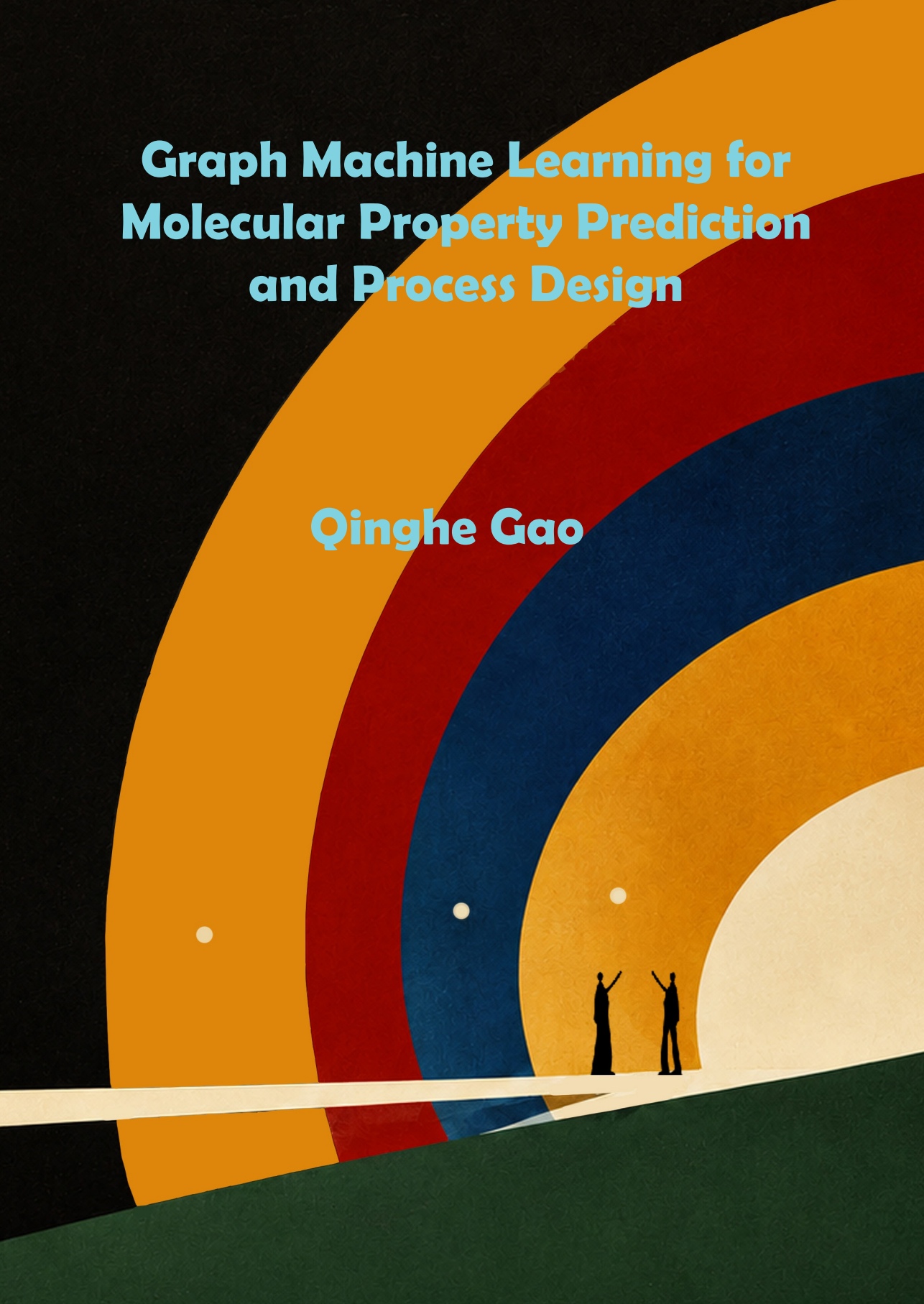
Graph Machine Learning for Molecular Property Prediction and Process Design

Qinghe Gao



Propositions

accompanying the dissertation

Graph Machine Learning for Molecular Property Prediction and Process Design

by

Qinghe Gao

1. For molecules and flowsheets, graph-based representations align more naturally with the underlying structure for machine learning.
   This proposition pertains to this dissertation.

2. Most of the research effort should go to data discovery, cleaning, and integration rather than to modeling in machine learning for chemical engineering.
   This proposition pertains to this dissertation.

3. Self-supervised learning cannot, by itself, solve label scarcity because it still requires a certain amount of high-quality labels.
   This proposition pertains to Chapter 5 of this dissertation.

4. Claims of reinforcement learning "inference" for process design remain unsubstantiated; credible, reproducible demonstrations are still lacking.
   This proposition pertains to Chapter 7 of this dissertation.

5. Closed ecosystems like Aspen Plus bottleneck genuine innovation in process design.

6. In academic discourse, the strongest critics of a new method are usually those least familiar with how it works.

7. Political small talk creates more friction than connection.

8. The single most important decision a PhD candidate makes is the choice of supervisor, not of research topic.

9. Mandatory holiday policies harm research productivity rather than help.

10. Society works hard to make introverts more talkative, and very little to make extroverts quieter.

These propositions are regarded as opposable and defendable, and have been approved as such by the promotor Prof. dr. ir. J. R. van Ommen and the copromotor Dr. A. M. Schweidtmann.

GRAPH MACHINE LEARNING FOR MOLECULAR PROPERTY PREDICTION AND PROCESS DESIGN

GRAPH MACHINE LEARNING FOR MOLECULAR PROPERTY PREDICTION AND PROCESS DESIGN

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
Friday 20 November 2026 at 10:00 a.m.

by

Qinghe GAO

The dissertation has been approved by the (co)promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof. dr. ir. J.R. van Ommen,	Delft University of Technology, promotor
Dr. A.M. Schweidtmann,	Delft University of Technology, copromotor
Independent Members:	
Prof. dr. ir. M.J.T. Reinders	Delft University of Technology
Prof. dr. ing. M.B. Franke	University of Twente
Dr. M.E. Leblebici	Katholieke Universiteit Leuven, Belgium
Dr. E. Isufi	Delft University of Technology
Prof. dr. ir. A.A. Kiss	Delft University of Technology, reserve member

This work was financially supported by the TU Delft AI Initiative and Erasmus+ Project.

TU Delft | AI Initiative

Key words: Graph machine learning, molecular property prediction, process design, reinforcement learning
Printed by: ProefschriftMaken
Cover by: Our Friday Goldfish
Copyright: 2026 by Q. Gao
ISBN (digital): 978-94-6518-488-3

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

Dedicated to the rocks I climbed and the places I traveled to

SUMMARY

Characterizing molecular properties and designing chemical processes are two essential tasks in chemical engineering. Both tasks rely on domain knowledge, empirical experience, and carefully engineered heuristics, and both have been accelerated in recent years by computer-aided methods, including group contribution, hierarchical decomposition, expert systems, and superstructure optimization. Deep learning methods now offer new opportunities, with promising applications already reported in property prediction, flowsheet synthesis, process control, and digitization.

While developing standalone deep learning tools for individual applications has produced promising results, these tools often remain limited in their utility for practical engineering work. A typical engineering workflow involves multiple tools and the input of multiple experts, including a property prediction tool to screen candidate molecules, a thermodynamic model to estimate phase behavior, a flowsheet simulator to evaluate process configurations, and a techno-economic or environmental assessment to score the result. Therefore, a centralized system that integrates multiple tools, databases, and experts, and that can iteratively exchange information and feedback across them, is needed to genuinely support the engineer throughout the whole workflow. The overarching objective of this dissertation is therefore twofold: first, to develop a set of deep learning tools that improve on existing methods for molecular property prediction and process design; and second, to identify a methodology to combine these tools, together with existing ones, into an integrated system that supports the engineer across the design workflow rather than at isolated steps within it.

Realizing this objective is constrained by five key challenges: (i) scarce labeled training data, (ii) the choice of data representation, (iii) over-confident predictions, (iv) optimal decision-making, and (v) integration of models. The dissertation argues that graphs are the appropriate representation on which all five challenges can be addressed coherently. Molecules are graphs of atoms and bonds; flowsheets are directed graphs of unit operations and streams; and the engineering workflow that coordinates them can itself be modeled as a graph of interacting agents and information flows. Graph-based models, such as graph neural networks, exploit this shared structure directly, respecting the locality, topology, and permutation invariance of chemical systems.

Building on this idea, the dissertation develops graph-based methods along three tracks: molecular property prediction, process design, and integration. On the molecular side, graph neural networks are shown to be a viable alternative to quantitative structure-property relationships in predicting environmental

impact categories with molecular structure information, although categories governed primarily by process and supply-chain factors remain beyond the reach of molecule-only models. Self-supervised pretraining at the node, edge, and graph level is introduced as a solution to label scarcity in polymer property prediction, improving R^2 by more than 30% under extreme data scarcity and revealing that the learned representation is distributed across both the encoder and the subsequent layers. Bayesian uncertainty quantification via Parallel Stochastic Gradient Hamiltonian Monte Carlo is developed as a scalable alternative to ensembles and Monte Carlo dropout, producing sharper and better-calibrated posteriors on thermodynamic benchmarks. On the process design side, flowsheet synthesis is formulated as a sequential decision problem on a directed graph and solved by a hierarchical reinforcement learning agent. Since learning efficiency is a bottleneck in this setting, two transfer strategies are investigated: pretraining on shortcut simulators before fine-tuning on rigorous ones, and initializing from the latent space of a process variational autoencoder. Pretraining on shortcut simulators and transferring the encoder and actor to the rigorous setting cuts convergence time by roughly 64% against the no-transfer baseline, whereas reusing continuous-action or critic networks across environments with mismatched reward shaping produces negative transfer. The results suggest that transfer learning serves as a carefully scoped accelerator for RL-based synthesis rather than a general remedy for learning inefficiency. Finally, multi-agent large language model systems are proposed as an orchestration layer above these specialized tools, surveying the required building blocks and outlining a concrete target for coupling molecular and process agents with embedded human oversight.

Returning to the five challenges, the contributions of this thesis collectively make tangible progress on each: graphs provide the relational representation, self-supervised pretraining and transfer learning mitigate data scarcity, Bayesian inference delivers calibrated uncertainty, hierarchical reinforcement learning addresses sequential decision-making, and multi-agent orchestration provides the integration layer. The overarching contribution is not any single method but the demonstration that graph-based learning, pursued consistently across scales, can serve as the connective tissue that begins to close the gap between molecular and process design.

Taken together, the contributions of this thesis suggest a research direction in which the three molecular models are integrated into a single general-purpose graph-based property predictor, in which reinforcement learning research for process design shifts its emphasis from optimization ability to inference generalization, and in which multi-agent orchestration provides the concrete mechanism by which molecular and process decisions are finally coupled into integrated design workflows. Looking further ahead, the thesis points toward a future in which

artificial intelligence is not an auxiliary tool invoked at specific steps of chemical engineering design but the substrate on which the workflow itself runs: molecular discovery, property prediction, process synthesis, and techno-economic and environmental assessment coupled on a shared platform that formulates tasks, selects among tools and databases, and consults human experts when needed. The role of the engineer correspondingly shifts from constructing models and executing simulations individually to specifying objectives, interrogating trade-offs, and adjudicating between candidate designs that the system itself has generated and evaluated.

SAMENVATTING

Het karakteriseren van moleculaire eigenschappen en het ontwerpen van chemische processen zijn twee essentiële taken binnen de chemische technologie. Beide taken steunen op domeinkennis, empirische ervaring en zorgvuldig ontwikkelde heuristieken, en beide zijn de afgelopen jaren versneld door computerondersteunde methoden, waaronder groepsbijdragemethoden, hiërarchische decompositie, expertsystemen en superstructuuroptimalisatie. Methoden uit deep learning bieden nu nieuwe mogelijkheden, met inmiddels veelbelovende toepassingen op het gebied van eigenschapsvoorspelling, flowsheetsynthese, procesregeling en digitalisering.

Hoewel de ontwikkeling van losstaande deep learning-tools voor afzonderlijke toepassingen veelbelovende resultaten oplevert, blijven deze tools in hun praktische bruikbaarheid voor het ingenieurswerk vaak beperkt. Een typische ingenieursworkflow omvat meerdere tools en de inbreng van meerdere experts, waaronder een tool voor eigenschapsvoorspelling om kandidaatmoleculen te screenen, een thermodynamisch model om fasegedrag te schatten, een flowsheetsimulator om procesconfiguraties te evalueren, en een techno-economische of milieukundige beoordeling om het resultaat te beoordelen. Daarom is een centraal systeem nodig dat meerdere tools, databases en experts integreert, en dat informatie en feedback iteratief tussen deze elementen kan uitwisselen, om de ingenieur werkelijk te ondersteunen gedurende de gehele workflow. De overkoepelende doelstelling van dit proefschrift is daarom tweeledig: ten eerste het ontwikkelen van een verzameling deep-learning tools die bestaande methoden voor moleculaire eigenschapsvoorspelling en procesontwerp verbeteren; en ten tweede het identificeren van een methodologie om deze tools, samen met bestaande tools, te combineren tot een geïntegreerd systeem dat de ingenieur ondersteunt over de gehele ontwerpworkflow in plaats van bij geïsoleerde stappen daarbinnen.

De realisatie van deze doelstelling wordt beperkt door vijf belangrijke uitdagingen: (i) schaarse gelabelde trainingsdata, (ii) de keuze van datarepresentatie, (iii) overconfidente voorspellingen, (iv) optimale besluitvorming, en (v) integratie van modellen. Het proefschrift betoogt dat grafen de geschikte representatie vormen waarop alle vijf uitdagingen samenhangend kunnen worden aangepakt. Moleculen zijn grafen van atomen en bindingen; flowsheets zijn gerichte grafen van procesonderdelen en stromen; en de ingenieursworkflow die deze coördineert, kan zelf worden gemodelleerd als een graaf van interagerende agents en informatiestromen. Graafgebaseerde modellen, zoals graph neural networks, benutten deze gedeelde structuur direct en respecteren daarbij de lokaliteit, topologie en permutatie-invariantie van chemische systemen.

Voortbouwend op dit idee ontwikkelt het proefschrift grafgebaseerde methoden langs drie sporen: voorspelling van moleculaire eigenschappen, procesontwerp en integratie. Aan de moleculaire kant wordt aangetoond dat graph neural networks een levensvatbaar alternatief zijn voor kwantitatieve structuureigenschapsrelaties bij het voorspellen van milieueffectcategorieën aan de hand van moleculaire structuurinformatie, hoewel categorieën die voornamelijk worden bepaald door proces- en toeleveringsketenfactoren buiten het bereik van uitsluitend moleculaire modellen blijven. Self-supervised voorafgaande training op knoop-, kant- en graafniveau wordt geïntroduceerd als oplossing voor labelschaarste bij de voorspelling van polymeereigenschappen, wat de R^2 met meer dan 30 % verbetert onder extreme schaarste en aantoont dat de geleerde representatie verdeeld is over zowel de encoder als de daaropvolgende lagen. Bayesiaanse onzekerheidskwantificering via Parallel Stochastic Gradient Hamiltonian Monte Carlo wordt ontwikkeld als een schaalbaar alternatief voor ensembles en Monte Carlo dropout, dat scherpere en beter gekalibreerde posterieuren oplevert op thermodynamische benchmarks. Aan de procesontwerpkant wordt flowsheetsynthese geformuleerd als een sequentieel beslissingsprobleem op een gerichte graaf en opgelost door een hiërarchische reinforcement learning-agent. Aangezien de leerefficiëntie in deze context een knelpunt vormt, worden twee transferstrategieën onderzocht: voorafgaande training op vereenvoudigde simulatoren gevolgd door bijstelling op rigoureuze simulatoren, en initialisatie vanuit de latente ruimte van een procesvariational autoencoder. Voorafgaande training op vereenvoudigde simulatoren gevolgd door overdracht van de encoder en actor naar de rigoureuze setting verkort de convergentietijd met ongeveer 64 % ten opzichte van de baseline zonder transfer, terwijl het hergebruiken van continuous-action- of critic-netwerken tussen omgevingen met afwijkende beloningsstructuren juist negatieve transfer oplevert. De resultaten suggereren dat transfer learning fungeert als een zorgvuldig afgebakende versneller voor RL-gebaseerde synthese in plaats van als een algemeen middel tegen leerinefficiëntie. Tot slot worden multi-agent large language model-systemen voorgesteld als een orchestratielaag boven deze gespecialiseerde tools, waarbij de benodigde bouwstenen worden geïnventariseerd en een concreet doel wordt geschetst voor het koppelen van moleculaire en procesagents met ingebed menselijk toezicht.

Terugkerend naar de vijf uitdagingen, leveren de bijdragen van dit proefschrift gezamenlijk tastbare vooruitgang op elk daarvan: grafen leveren de relationele representatie, self-supervised voorafgaande training en transfer learning verzachten de dataschaarste, Bayesiaanse inferentie levert gekalibreerde onzekerheid op, hiërarchische reinforcement learning richt zich op sequentiële besluitvorming, en multi-agent orchestratie biedt de integratielaag. De overkoepelende bijdrage is geen enkele afzonderlijke methode, maar de demonstratie dat graafgebaseerd

leren, consistent toegepast over verschillende schalen, kan dienen als het verbindende weefsel dat de kloof tussen moleculair ontwerp en procesontwerp begint te overbruggen.

Samengenomen wijzen de bijdragen van dit proefschrift in de richting van een onderzoeksrichting waarin de drie moleculaire modellen worden geïntegreerd tot één algemene graafgebaseerde eigenschapsvoorspeller, waarin onderzoek naar reinforcement learning voor procesontwerp zijn nadruk verplaatst van optimalisatievermogen naar generalisatievermogen bij inferentie, en waarin multi-agent orchestratie het concrete mechanisme vormt waarmee moleculaire en procesgerelateerde beslissingen uiteindelijk worden gekoppeld in geïntegreerde ontwerpworkflows. Vooruitkijkend wijst het proefschrift op een toekomst waarin kunstmatige intelligentie geen hulpmiddel meer is dat bij specifieke stappen van het chemisch-technisch ontwerp wordt ingeschakeld, maar de basis waarop de workflow zelf draait: moleculaire ontdekking, eigenschapsvoorspelling, processynthese en techno-economische en milieukundige beoordeling, gekoppeld op een gedeeld platform dat taken formuleert, kiest tussen tools en databases, en menselijke experts raadpleegt wanneer dat nodig is. De rol van de ingenieur verschuift dienovereenkomstig van het zelfstandig bouwen van modellen en uitvoeren van simulaties naar het specificeren van doelstellingen, het bevragen van afwegingen, en het beoordelen tussen kandidaat-ontwerpen die het systeem zelf heeft gegenereerd en geëvalueerd.

PREFACE

As I once read, "There is no concept called time; you only remember or realize it when something brings you joy, sadness, excitement, or fear." Time, in essence, is woven by the intensity of our experiences. It is these moments—whether clear or hazy in memory—that define our perception of time.

And that is why I still remember it all too well: the story began with one email. It was 19:41 in the evening when I came across a PhD position that caught my attention, prompting me to send an inquiry to the professor. Honestly, I didn't expect much—this email, like so many before it, would likely vanish into the sea of unread messages, never to receive a reply. But at 19:58, the familiar sound of a Gmail notification broke the silence. Artur had replied. I was stunned, and with that single, unexpected response, a five-year journey began.

About two years into my PhD, I remember asking everyone in our office, "If you could go back to any period of your studies, when would you choose?" Most people fondly recalled their high school or bachelor's years. For them, those were the times of discovery and freedom. But, unlike them, I had a different answer. With a certainty I can still feel, I replied, "For me, it is now—this PhD time." And now, standing at the edge of completion, I find myself looking back and realizing that my answer hasn't changed. I can still say, "Yes, it is my PhD life."

Qinghe Gao
Delft, May 2026

CONTENTS

Summary vii

Samenvatting xi

Preface xv

I Introduction 1

1 Introduction 3
1.1 A graph view of molecules, flowsheets, and engineering workflow . . 5
1.2 Learning on graphs: from message passing to fingerprints 7
1.3 Graph machine learning in chemical engineering 7
1.4 Scope and limitations 9
1.5 Organization of the thesis 10
References 10

II Background 13

2 GNNs for the molecular property predictions 15
2.1 Introduction 16
2.2 Background 17
2.3 Graph neural networks for molecular property prediction 19
2.4 Numerical examples 27
2.5 Concluding remarks 33
References 34

3 Deep RL for process design: Review and perspective 39
3.1 Introduction 40
3.2 State of the art 43
3.3 Perspectives. 48
3.4 Conclusions. 52
References 52

III Graph machine learning for molecular property prediction 57

4 Environmental impacts prediction using GNNs 59
4.1 Introduction 60
4.2 Preliminaries 64
4.3 Datasets 66
4.4 Methods. 69

4.5 Results and discussion . . . 75
4.6 Conclusion . . . 84
4.7 Acknowledgements. . . . 87
4.8 Declaration of generative AI and AI-assisted technologies in the writing process . . . 87
References . . . 87
4.9 Appendix . . . 93

5 Self-supervised GNNs for polymer property prediction 99
5.1 Introduction . . . 100
5.2 Methods. . . . 102
5.3 Results and discussion . . . 111
5.4 Conclusions. . . . 115
References . . . 118

6 Bayesian uncertainty quantification of GNNs using SGHMC 121
6.1 Introduction . . . 122
6.2 Methods. . . . 123
6.3 Results and discussion . . . 126
6.4 Conclusions. . . . 128
References . . . 128

IV Graph machine learning for process design 131

7 Transfer learning for process synthesis with RL 133
7.1 Introduction . . . 134
7.2 RL for process synthesis . . . 136
7.3 Case study . . . 142
7.4 Results and discussion . . . 145
7.5 Conclusions. . . . 158
7.6 Acknowledgements. . . . 159
References . . . 159

V Integration of molecular and process design 163

8 Multi-agent systems for chemical engineering 165
8.1 Introduction . . . 166
8.2 State of the art . . . 167
8.3 Perspective . . . 174
8.4 Conclusions. . . . 180
8.5 Acknowledgement . . . 181
References . . . 181

VI Summary and looking ahead **187**

9 Conclusions and Outlook **189**
9.1 On the suitability of graph representations. 189
9.2 Graph machine learning for molecular properties. 190
9.3 Graph machine learning for process design 191
9.4 Graph machine learning for orchestration 191
9.5 Returning to the two objectives and five challenges 192
9.6 Limitations 192
9.7 Outlook 194

Acknowledgements **197**

Curriculum Vitæ **203**

Supervision Activities **205**

List of Publications **207**

Propositions **209**

I

INTRODUCTION

1 INTRODUCTION

Characterizing molecular properties and designing chemical processes are two essential tasks in chemical engineering. The first asks how a molecule behaves: its reactivity, its environmental impact, and its electronic structure. The second asks how a configuration of unit operations can convert feedstocks into products at the required purity, yield, and cost. In practice, the development workflow for solving these two tasks requires extensive domain knowledge, a deep understanding of context, and complex heuristics. Consequently, many computer-aided approaches have been developed to speed up these workflows. For example, computational methods such as group contribution methods [1] and quantitative structure-property/-activity relationships (QSPR/QSAR) [2] have been used for property modeling. Similarly, hierarchical decompositions [3], expert systems [4], and superstructure optimization [5] have been deployed to accelerate conceptual process design.

More recently, deep-learning-based methods have transformed prediction and design across many domains [6–8]. In chemical engineering, these approaches have shown promising results in control [9], digitization [10], and property prediction [11]. While developing standalone deep learning tools for individual applications has produced promising results, those tools remain limited in their utility for practical engineering work. A typical engineering workflow involves multiple tools and the input of multiple experts, including a property prediction tool to screen candidate molecules, a thermodynamic model to estimate phase behavior, a flowsheet simulator to evaluate process configurations, and a techno-economic or environmental assessment to score the result. Therefore, a centralized system that integrates multiple tools, databases, and experts, and that can iteratively exchange information and feedback across them, is needed to genuinely support the engineer throughout the whole workflow.

The overarching objective of this dissertation is twofold. The first is to develop a set of deep learning tools that improve on existing methods for molecular property prediction and process design. The second is to compose these tools, together with existing ones, into an integrated system that supports the engineer across the design workflow rather than at isolated steps within it.

Realizing this objective is constrained by five challenges. First, labeled data is scarce. Unlike computer vision or natural language processing, where datasets

consisting of millions of data points are readily available, chemical datasets are costly to generate: every measured polymer property, every rigorously simulated flowsheet, every life-cycle inventory entry requires laboratory experiments or high-fidelity simulation. As a result, curated datasets in this field typically span hundreds to a few thousand labeled examples, rather than the millions common in adjacent domains. Models trained on such datasets from random initialization tend to overfit and generalize poorly beyond the training distribution. Progress, therefore, depends on methods that can learn from limited labels. Self-supervised pretraining, transfer across related tasks, and inductive biases grounded in molecular and process structure are the principal tools available.

Second, the choice of data representation is important. Molecules and processes are structurally complex objects, carrying information not only in their atoms, bonds, unit operations, and streams, but also in the topology, directionality, stereochemistry, and recycle structures that connect them. Classical representations such as molecular fingerprints, SMILES strings, or tabular descriptors of process variables cannot accommodate this richness, which generally flattens the underlying structure into fixed-length vectors or sequences, and much of the topological and relational information is often lost in the encoding. An ideal representation should instead align with the natural structure of the system, preserving connectivity, directionality, and relational information. Such a representation provides a stronger foundation for learning, since the model can focus its capacity on the physics and chemistry of interest rather than on reconstructing structural relationships that were already present in the raw data. Beyond individual models, a representation that supports the integration of multiple models is also critical, since coordinating information across specialized tools, databases, and human experts is a demanding task.

Third, a prediction without a calibrated uncertainty estimate is of limited practical use. Uncertainty quantification is vital for the deployment of a machine learning model, and it becomes especially acute in chemical engineering, where predictions drive decisions with technical, economic, and safety consequences. The problem becomes especially severe in extrapolative regimes, where training data is sparse and where prediction errors compound as outputs propagate into downstream design decisions. Principled uncertainty quantification, which separates irreducible aleatoric noise from reducible epistemic model uncertainty, is therefore not a reporting convenience but a precondition for deploying learned models in any decision-critical setting.

Fourth, optimal decision-making is difficult. Many chemical engineering tasks, including process synthesis, reaction planning, and experimental design, are not regression problems but sequential, combinatorial searches over constrained design spaces. In current practice, such tasks rely on predefined super-

structures, hand-crafted heuristics, and exhaustive simulation, all of which are time-consuming and depend heavily on the expertise of the engineer driving them. A learned model that could propose promising candidates directly, learn from the outcomes of past decisions, and exploit the structural regularities of the design space would accelerate these workflows and provide decision support.

Finally, integration and coordination of models is critical. Current engineering practice relies on a heterogeneous collection of specialized tools and models, for example, property prediction, flowsheet simulator, techno-economic analysis, and so on. A system that integrates these individual models, routing information between them and composing their outputs into coherent workflows, would unlock design capabilities that no individual model currently provides. Building such a system demands an orchestration layer capable of reasoning about when each model is applicable and of coordinating their outputs into end-to-end design decisions.

Addressing these five challenges in a unified way requires a representation that captures the heterogeneous structure of chemical systems to support learning and decision-making. The central idea of this dissertation is that graphs provide that representation. Molecules are graphs of atoms and bonds, and flowsheets are directed graphs of unit operations and streams. As the final part of this thesis argues, the engineering workflow that coordinates them can itself be modeled as a graph of interacting agents and information flows. This shared structure is what allows graph neural networks (GNNs) [12] to learn structure–property relations at the molecular scale, to inform sequential decisions at the process scale, and to be composed into systems that span the two.

1.1. A GRAPH VIEW OF MOLECULES, FLOWSHEETS, AND ENGINEERING WORKFLOW

Formally, a graph is defined as $G = (V, E)$, comprising a set of nodes (or vertices) $v \in V$ and a set of edges $e_{vw} \in E$ connecting nodes v and w. In the context of deep learning, these structures are typically *attributed graphs*, meaning they carry information beyond connectivity. Mathematically, we assign a node v with $\mathbf{f}^v \in F^V$ and a feature vector of an edge e_{vw} with $\mathbf{f}^{e_{vw}} \in F^E$.

This mathematical framework maps naturally to chemical engineering systems. As Figure 1.1 shows, for molecules, nodes v represent atoms, where $\mathbf{f}^v$ encodes features such as atomic number, valence, and hybridization. Edges e_{vw} represent bonds, where $\mathbf{f}^{e_{vw}}$ encodes bond order, aromaticity, and interatomic distances. For processes, the abstraction scales up: nodes represent unit operations defined by design and operating variables, while edges represent material or energy streams characterized by flow rate, temperature, pressure, and composition.

1

The same abstraction extends one level further, to the engineering workflow that coordinates molecular and process tools. Here, nodes represent specialized agents (powered by large language models), human experts, or tools, and edges represent the structured exchanges of information between them, including queries, results, uncertainty estimates, and human interventions. While the methods applied at this third level differ from those used at the molecular and process scales, the underlying graph abstraction remains consistent and provides a unified language in which the integration of tools can be described and reasoned about.

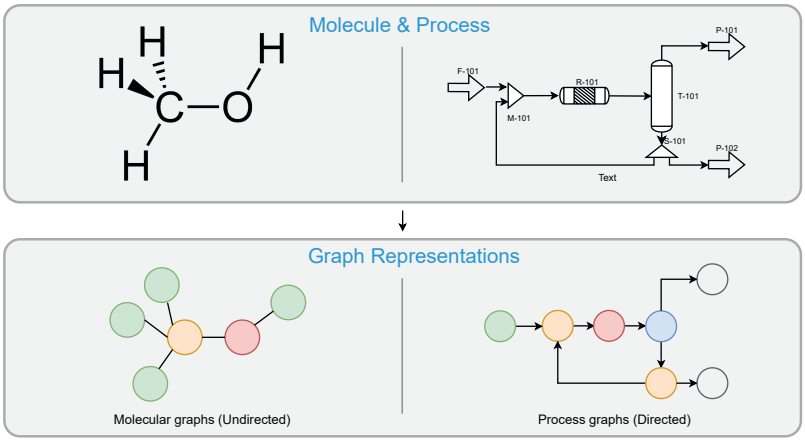


Figure 1.1: Graph representation of chemical systems. (Left) A molecular graph where nodes represent atoms and edges represent bonds. (Right) A process flowsheet where nodes represent unit operations and edges represent material/energy streams. The same abstraction extends to the engineering workflow that coordinates molecular and process tools, as discussed in the text and developed in Part V.

While these domains share this underlying graph structure, they exhibit distinct topological properties. Molecular graphs are typically treated as *undirected* regarding connectivity, whereas flowsheet graphs are inherently *directed*, governed by stream orientation and mass–energy balances. Workflow graphs may be either directed or undirected: information often flows from one agent to the next in a defined order, but feedback loops, human revisions, and iterative tool calls can introduce bidirectional exchanges. This distinction significantly influences information flow in model architectures. Practical encodings for these features follow standard conventions: Categorical attributes are often processed via one-hot encoding or learnable embeddings, while continuous variables are numerically scaled. Ultimately, this shared representation allows properties and decisions to be posed in a single modeling language while preserving the unique

domain-specific physics implied by each graph.

1.2. LEARNING ON GRAPHS: FROM MESSAGE PASSING TO FINGERPRINTS

GNNs rely on a mechanism known as message passing to learn from these graph structures. In this process, nodes iteratively exchange and aggregate information with their neighbors to update their internal features. Through distinct layers of interaction, a node gradually accumulates knowledge not just about its immediate chemistry or unit operation settings, but also about the broader topology of the system. Architectural variants such as graph convolution, attention mechanisms, and geometry-aware layers have been developed to refine this process, addressing specific challenges like long-range dependencies or 3D spatial effects. Once this local structure is encoded, a pooling or readout phase maps the node-level representations to a single graph-level vector. This resulting vector acts as a learned "fingerprint" of the molecule or flowsheet, compacting complex structural information into a format suitable for task-specific predictors. Finally, the resulting vector is mapped into the target prediction by a multilayer perceptron.

1.3. GRAPH MACHINE LEARNING IN CHEMICAL ENGINEERING

Building on this foundation, this dissertation introduces a graph-based methodology that addresses molecular property prediction and sequential process design and proposes their integration through multi-agent orchestration.

To mitigate data scarcity, self-supervised pretraining strategies are developed for molecular and polymer graphs, allowing models to exploit large pools of unlabeled structures before fine-tuning on small labeled targets such as electron affinity and ionization potential. To deliver calibrated uncertainty, directed message-passing encoders are augmented with mean–variance outputs for aleatoric noise, and Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) is adopted to sample weight posteriors for epistemic uncertainty, so that each prediction is accompanied by an estimate of both the noise in the data and the uncertainty of the model. To address design as sequential decision-making, flowsheet synthesis is formulated as a Markov decision process on a directed graph, with a hierarchical reinforcement learning agent that proposes units and connections under thermodynamic and topological constraints, supported by transfer learning to contain simulator cost. Finally, to integrate molecular and process decisions, the preceding models are exposed as executable tools within a multi-agent large language model framework, in which a supervisor agent decomposes design queries, routes information between molecular and process models, and interprets their outputs as a coherent end-to-end workflow. These methods do not replace rigorous first-

principles simulation or deterministic optimization. Instead, they complement them by narrowing the search space, quantifying where model predictions can be trusted, and coordinating specialized models into workflows that neither tool alone can support. The remainder of this chapter situates these contributions within the broader landscape of graph machine learning in chemical engineering and outlines the organization of the thesis.

1.3.1. GRAPH LEARNING FOR MOLECULES (PART III)

This part targets property prediction with environmental relevance and robustness, beginning with GNNs that predict multiple impact categories directly from molecular graphs to inform LCA-aware screening. Performance is evaluated against QSPR/QSAR baselines and modern benchmarks, clarifying when message passing, geometric inputs, or multitask formulations add value. Next, self-supervised pretraining at node, edge, and graph levels is explored to address data scarcity in polymers before fine-tuning to specific targets such as electron affinity and ionization potential, improving sample efficiency and generalization. Furthermore, directed message passing encoders are augmented with mean–variance outputs for aleatoric uncertainty, $\mu(x)$ and $\sigma^2(x)$, and Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) is adopted to sample weight posteriors for epistemic uncertainty, and its calibration is benchmarked against Deep Ensembles and MC Dropout.

1.3.2. GRAPH LEARNING FOR PROCESS DESIGN (PART IV)

The part formulates flowsheet synthesis as a sequential decision-making problem on a directed graph. In this framework, the state represents a partially built process, and the agent's actions, such as adding, connecting, and sizing units, are executed in a hierarchical, hybrid manner using deep reinforcement learning (RL). While RL is effective for goal-directed learning through agent-environment interactions, a major challenge arises from its data inefficiency: the learning process demands a vast number of queries to rigorous process simulators, resulting in prohibitive simulation times and computational costs. To mitigate these simulator bottlenecks, the agent leverages multi-fidelity surrogates and transfer learning, pretraining on computationally cheaper models before fine-tuning on rigorous simulators. Furthermore, generative latent models, such as variational autoencoders (VAEs), are employed to provide structure-aware priors that focus exploration. Ultimately, this integration reduces the number of expensive evaluations required to reach high-quality designs.

1.3.3. ORCHESTRATION OVER MODELS (PART V)

Finally, this part presents a perspective on orchestrating the developed graph learning frameworks via a tool-augmented, multi-agent Large Language Model (LLM) system. In this vision, the specialized models developed in the previous parts are encapsulated as executable tools that LLM-based agents can invoke. Rather than relying on hand-coded pipelines, the agents dynamically invoke the uncertainty-aware GNNs to screen molecular candidates and trigger the hierarchical RL strategies to generate corresponding flowsheets. This framework models the engineering workflow as a *conversation graph*, where a supervisor agent decomposes complex design queries, routes data between the molecular and process models, and interprets their outputs. Crucially, this orchestration allows for logic-driven data flow: for instance, the LLM can use the calibrated uncertainty estimates to decide whether to query the process design agent for a full design or to request additional data, providing a concrete mechanism by which the loop between molecular discovery and process synthesis can begin to be closed.

1.4. SCOPE AND LIMITATIONS

The computational frameworks proposed herein are designed to complement, rather than replace, rigorous first-principles simulation and deterministic optimization. While the graph-based models accelerate screening and propose high-probability designs, the final validation must still rely on detailed thermodynamic and kinetic modeling. Furthermore, the deployment of GNN-based molecular property models remains largely academic. Bridging the gap to industrial usage requires overcoming significant challenges regarding interpretability, data scarcity in proprietary domains, and trust compared to established empirical correlations. Moreover, RL is primarily utilized for dynamic control problems or sequential games. In contrast, stationary process synthesis is inherently a static optimization problem, where the objective is to find a single optimal topology and operating point. Mapping this static problem into a sequential construction process introduces complexities in reward shaping and state representation. Consequently, the comparative efficacy of RL against established meta-heuristics (such as Genetic Algorithms) or deterministic approaches (Superstructure Optimization/MINLP) remains as an open question, specifically regarding sample efficiency and convergence guarantees.

In addition, the inference or generalization capability of the RL agent presents a scope for further inquiry. While the agent learns to design flowsheets within the training distribution, its ability to extrapolate to unseen design specifications, novel unit operations, or distinct chemical systems (without retraining) remains an open challenge that this work begins to characterize but does not fully solve.

Finally, the strength of the empirical evidence differs between chapters, and some of the studies are based on limited benchmarks. A critical reflection on this point is given in Part VI.

1.5. ORGANIZATION OF THE THESIS

Part I introduces the motivation, approach, and thesis layout, framing graphs as the connective tissue between properties and design. **Part II** provides background on graph neural networks for molecular structure–property prediction and deep reinforcement learning for process design, establishing the methodological toolkit. Chapter 2 in this part is reproduced from a book chapter led by J. G. Rittig and is included as background only. It is not claimed as an original contribution of this thesis, and the precise contribution of the author is stated at the beginning of the chapter. **Part III** develops graph learning for molecules, covering environmental impact prediction, self-supervised GNNs for polymers, and Bayesian uncertainty quantification via SGHMC. **Part IV** develops graph learning for process design, formulating flowsheet synthesis as sequential decisions on directed graphs with hierarchical RL, transfer across fidelities, and VAE-guided exploration. **Part V** introduces a perspective on leveraging multi-agent LLMs to orchestrate these graph machine learning models, unifying the molecular and process scales. The chapter in this part is a review and perspective led by S. Rupprecht, and the contribution of the author is stated at the beginning of the chapter. Finally, **Part VI** summarizes the contributions, returns to the five challenges introduced in this chapter, discusses limitations, and outlines the open questions and broader research directions to which this work points.

REFERENCES

[1] R. Gani, *Group contribution-based property estimation methods: advances and perspectives,* Current Opinion in Chemical Engineering **23**, 184 (2019).

[2] L. P. Hammett, *Some relations between reaction rates and equilibrium constants.* Chemical Reviews **17**, 125 (1935).

[3] M. M. Daichendt and I. E. Grossmann, *Integration of hierarchical decomposition and mathematical programming for the synthesis of process flowsheets,* Computers & Chemical Engineering **22**, 147 (1998).

[4] S.-H. Liao, *Expert system methodologies and applications—a decade review from 1995 to 2004,* Expert Systems with Applications **28**, 93 (2005).

[5] L. Mencarelli, Q. Chen, A. Pagot, and I. E. Grossmann, *A review on superstructure optimization approaches in process system engineering,* Computers & Chemical Engineering **136**, 106808 (2020).

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks,* Communications of the ACM **60**, 84 (2017).

[7] S. Chithrananda, G. Grand, and B. Ramsundar, *Chemberta: Large-scale self-supervised pretraining for molecular property prediction,* (2020), 10.48550/ARXIV.2010.09885, arXiv:2010.09885 [cs.LG] .

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need,* (2017), 10.48550/ARXIV.1706.03762, arXiv:1706.03762 [cs.CL] .

[9] D. B. Raven, Y. Chikkula, K. M. Patel, A. H. Al Ghazal, H. S. Salloum, A. S. Bakhurji, and R. S. Patwardhan, *Machine learning & conventional approaches to process control & optimization: Industrial applications & perspectives,* Computers & Chemical Engineering **189**, 108789 (2024).

[10] M. F. Theisen, K. N. Flores, L. Schulze Balhorn, and A. M. Schweidtmann, *Digitization of chemical process flow diagrams using deep convolutional neural networks,* Digital Chemical Engineering **6**, 100072 (2023).

[11] W. P. Walters and R. Barzilay, *Applications of deep learning in molecule generation and molecular property prediction,* Accounts of Chemical Research **54**, 263 (2020).

[12] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, *The graph neural network model,* IEEE Transactions on Neural Networks **20**, 61 (2009).

II

BACKGROUND

2

GRAPH NEURAL NETWORKS FOR THE PREDICTION OF MOLECULAR STRUCTURE–PROPERTY RELATIONSHIPS

Molecular property prediction is of crucial importance in many disciplines such as drug discovery, molecular biology, or material and process design. The frequently employed quantitative structure-property/activity relationships (QSPRs/QSARs) characterize molecules by descriptors which are then mapped to the properties of interest via a linear or nonlinear model. In contrast, graph neural networks, a novel machine learning method, directly work on the molecular graph, i.e., a graph representation where atoms correspond to nodes and bonds correspond to edges. GNNs allow to learn properties in an end-to-end fashion, thereby avoiding the need for informative descriptors as in QSPRs/QSARs. GNNs have been shown to achieve state-of-the-art prediction performance on various property predictions tasks and represent an active field of research. We describe the fundamentals of GNNs and demonstrate the application of GNNs via two examples for molecular property prediction.

This chapter has been published as " Machine Learning and Hybrid Modelling for Reaction Engineering Theory and Applications" by Rittig, J. G., Gao, Q., Dahmen, M., Mitsos, A., & Schweidtmann, A. M., Royal Society of Chemistry 2023, vol. 26, ch. 7, pp. 159-181

Note on the contribution of this chapter. This chapter is included as background on graph neural networks for molecular property prediction and is not claimed as an original scientific contribution of this dissertation. The work was led by J. G. Rittig as first author, who has also included it in his doctoral dissertation at RWTH Aachen University. The contribution of the author of this dissertation was limited to drafting the Introduction, reviewing, and editing the manuscript.

2.1. INTRODUCTION

The estimation of molecular properties is a critical task in many disciplines such as drug discovery, molecular biology, or material and process design. Identification and evaluation of new molecules were originally manual, intuition-driven processes performed by scientists. Since the 1930s, a mathematical modeling approach, the so-called quantitative structure-property relationship (QSPR) [1], has provided an effective and efficient framework to screen and explore the chemical search space by computational means. QSPRs characterize molecules with various structural, chemical, physical, and biological features, referred to as molecular descriptors, which are then mapped to a property of interest by a linear or nonlinear model. However, the selection of informative descriptors is a difficult task that typically requires domain knowledge and expert intuition, with a bad choice leading to poor prediction performance.

Recently, advances in computational hardware and software have led to the emergence of deep learning, including novel machine learning (ML) approaches that have achieved immense success in several fields such as pattern recognition and natural language processing [2]. Rather than relying on expert intuition for feature engineering, deep learning is capable of automatically learning underlying representations from data. This has also led to a resurgence and further developments in the field of ML-based property prediction and molecule generation [3–5]. In particular, graph neural networks (GNNs) have shown promising results [4, 5], as they can learn relevant features directly from the molecular graph, i.e., an intuitive representation of molecules where atoms and bonds corresponds to nodes and edges of a graph, respectively, through supervised end-to-end training.

GNNs reached state-of-the-art accuracy in predicting molecular properties, for both regression tasks, like dipole moment or enthalpy of atomization, e.g., in [4, 6], and classification tasks, such as toxicity of molecules, e.g., in [7]. Further refinement of GNN architectures (cf. [8, 9]) have led to increasing prediction accuracies on several benchmark data sets, e.g., ZINC 15 [10] or QM9 [11, 12].

High interest by the computer science and applied science community currently catalyzes the development of new GNN approaches and applications. For example, we have developed higher-order GNNs [13] and applied these to the prediction of fuel auto-ignition quality [14]. GNNs have also been applied to the prediction of activity coefficients, an important problem in chemical engineering [15]. This rising attention shows analogies to fields in which ML models have shown remarkable achievements such as convolutional neural networks (CNNs) for image recognition or transformer models for natural language processing.

In this chapter, we provide an introduction to GNNs for end-to-end learning of structure-property relationships of molecules, encouraging the application of GNNs for broader use in chemical engineering. As a starting point, we briefly

outline the classical QSPR concept (Section 2.2.1) and review developments leading to ML-based molecular graph approaches (Section 2.2.2). We then describe the concept of GNNs for molecular property prediction in detail (Section 2.3). Two examples are presented applying GNNs for molecular property prediction in regression (Section 2.4.1) and classification tasks (Section 2.4.2). We end the chapter with a short conclusion, a brief overview of recent GNN developments, and an outlook (Section 2.5).

2.2. BACKGROUND

We first introduce QSPR modeling, an established and frequently used technique for molecular property prediction, and then present neural molecular fingerprints as a first step towards end-to-end learning of properties from molecular structure.

2.2.1. QUANTITATIVE STRUCTURE-PROPERTY RELATIONSHIPS

QSPR modeling is a two-step process consisting of (i) molecule description and (ii) property regression. In step (i), the structure of a molecule m is utilized to compute so-called molecular descriptors d_i in a mapping $D : m \mapsto \mathbf{d}$ with $\mathbf{d} = [d_1, d_2, ..., d_n]^T$. Note that sometimes also experimentally determined data is used as descriptor data. In the subsequent step (ii), the descriptors are correlated with the property $\hat{p}$ to be predicted [16] with the help of a function $F(\cdot)$, i.e.,

$$\hat{p} = F(\mathbf{d}) = F(D(m)). \tag{2.1}$$

Besides linear and nonlinear (multivariate) regression approaches, methods from ML such as random forests, support vector machines, or artificial neural networks are commonly employed for predicting the molecular property $\hat{p}$.

Different types of molecular descriptors $\mathbf{d}$ can be distinguished according to their dimensionality[17, 18]: *0D-descriptors* are molecular features for which no information about the structure and connectivity between atoms is required, e.g., the count of each atom type in a molecule. So-called *1D-descriptors* describe features derived from subgroups or fragments within a molecule, e.g., the count of functional groups. *2D-descriptors* refer to the topology of a molecule and describe atoms and their bonds through a molecular graph including additional graph features like symmetry, branching, and cyclicity. Lastly, *3D-* or *higher-dimensional descriptors* exploit the molecular geometry, molecular interaction fields, and time-dependent molecular dynamics. For a comprehensive list of molecular descriptors, the interested reader is referred to the literature [18, 19].

Two frequently used descriptor types are structural groups (2D) and molecular fingerprints (2D and/or 3D). Group contribution methods (GCMs) [20, 21], also referred to as group additivity methods, decompose the molecular structure into

predefined structural groups, e.g., >CH- (non-ring), =CH- (ring), or -OH (non-ring), with the number of occurrence of these groups constituting the molecular descriptors. Molecular fingerprints, on the other hand, represent molecules as vectors in which feature information about the molecular structure is stored, typically in the form of binary or integer values [18, 19]. Features are, e.g., the count of substructures, similar to the GCM, or geometric distances between two atoms or structural groups [18].

A particular challenge in QSPR modeling is the choice of suitable descriptors for a specific property prediction task at hand, as only informative descriptors may facilitate successful property prediction [17]. Often, however, informative descriptors are not known *a priori*. Both trial-and-error descriptor selection by the modeler and automated descriptor selection strategies can lead to chance correlations and therefore spurious results, particularly on small data sets. Creating a predictive QSPR model therefore may require expert intuition and must be based on a careful selection of data and algorithms for modeling and validation [16, 17].

2.2.2. NEURAL MOLECULAR FINGERPRINTS

Molecular fingerprints have long been utilized to describe molecular structures by a vector representation, with the neural molecular fingerprint being a particular type of fingerprint that bears similarities with GNNs. Neural molecular fingerprints are based on the concept of circular molecular fingerprints [22, 23] that apply a neighborhood aggregation scheme to learn the local environment of an atom, often depicted as circle and illustrated in Figure 2.1. Each atom within a molecule is considered individually and information of its neighboring atoms is aggregated and stored in an atom feature vector [24]. Subsequently, the concatenation of the feature vector of an atom with the feature vectors of the neighboring atoms is mapped to a single-integer identifier by applying a hash function [24]. The neighborhood aggregation process is repeated for a pre-defined number of iterations, referred to as radius R. As shown in Figure 2.1, the radius corresponds to the size of the neighborhood from which a node under consideration receives information. In the first iteration, $R = 1$, only information from the direct neighboring atoms can be retrieved. In the next iteration, $R = 2$, still only direct neighboring atoms pass information to each other. However, since all atoms have already received information about their direct neighboring atoms in the previous iteration, they can now pass on this information. For $R = 3$, the information radius increases again, and so on. Note that the radius is also referred to as the R-hop environment (cf. Section 2.3.2). The learned local atom environments are then hashed to an integer value which is associated with a binary encoding in the final circular fingerprint representations. Thus, circular fingerprint methods result in a binary fingerprint vector that encodes molecular structure information. Popular

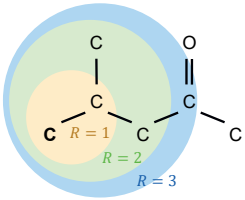


Figure 2.1: Neighborhood aggregation scheme for one atom (bold C atom at lower left). R indicates the radius, i.e., the size of the neighborhood. Illustrative example for 4-methyl-2-pentanone.

Duvenaud et al. (2015) [25] significantly advanced the generation of circular molecular fingerprints by modifying the neighborhood aggregation scheme to be differentiable, thereby introducing the concept of neural molecular fingerprints. Instead of a hash function, neural molecular fingerprints apply a smooth activation function σ (which is similar to the activation function in neural networks) for updating the atom feature vectors. Specifically, a softmax function is used for σ such that the atom feature vector entries become real-valued resulting in a continuous atom vector representation. In addition, the aggregation of neighbors is adjusted to be size-agnostic regarding the number of atoms by applying the sum operator instead of concatenating the neighbor feature vectors. The continuous atom feature vectors are then combined into a real-valued fingerprint vector, referred to as the neural molecular fingerprint [25]. Further, Duvenaud et al. (2015) [25] introduced learnable parameters in the neighborhood aggregation scheme, thus allowing to train the differentiable neural molecular fingerprint with respect to specific property prediction tasks. Neural molecular fingerprints have been shown to outperform circular fingerprints on several prediction tasks, like solubility or drug efficacy [25], and have paved the way for direct learning from molecular structures.

2.3. Graph neural networks for molecular property prediction

GNNs are a deep learning method capable of learning properties from graph representations, thereby enabling end-to-end learning from molecular graphs to properties and eliminating the need for informative descriptors that is inherent to QSPR/QSAR modeling.

As GNNs operate on graphs, a representation of nodes and edges used to model objects and their relationships, they correspond to the field of ML within

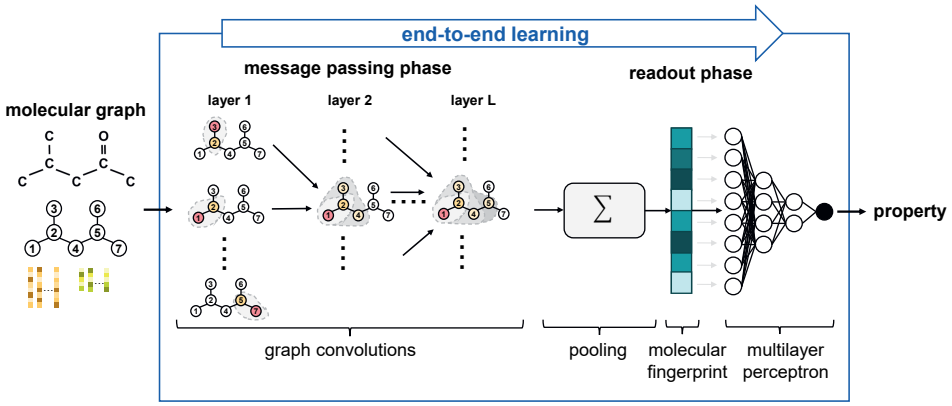


Figure 2.2: Overview of a GNN model for property prediction, similar to the GNN we used for predicting fuel ignition qualities of hydrocarbons, cf. [14].

the non-Euclidian data domain (e.g., grids, graphs, and groups), referred to as geometric deep learning [26]. The concept of GNNs was introduced by Gori et al. [27] and Scarselli et al. [28]: Based on a graph as input, parametric mathematical operations are applied to learn graph characteristics that can be correlated to properties attributed to one specific node, a subregion of the graph (subgraph), or the entire graph. As GNNs can be trained with backpropagation methods from ML, they enable end-to-end learning from the graph to the property of interest.

GNNs learn graph properties by aggregating the characteristics of local graph regions into an overall graph representation. Their learning concept is similar to that of convolutional neural networks (CNNs), one of the most successful achievements in ML allowing to recognize objects in images [2]. CNNs learn by first decomposing the pixel grid into many small, local grids to extract fine-grained features and then step-wise aggregating those local grids to reassemble fine-grained features into overall characteristics of an image. Graphs, however, are size-agnostic, i.e., the number of nodes may vary between different samples, and permutation (or isomorphism) invariant, i.e., the nodes of a graph do not have a specific order. GNNs are designed to address these special characteristics, i.e., they work on graphs with varying input sizes and their output does not depend on the input order of the graph nodes. GNNs learn graph features by considering each node and its neighborhood individually, hence local graph regions, and then aggregating the single node representations into an overall graph representation.

Recently, GNNs have emerged for molecular property prediction [29] from molecular graphs, a graph representation of molecules where atoms and bonds correspond to nodes and edges, respectively. For molecular property prediction, typically, GNNs in the form of message passing neural networks (MPNNs) [4]

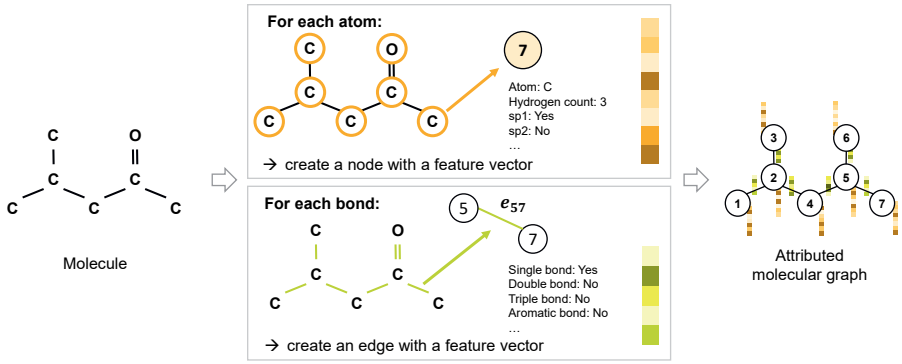


Figure 2.3: Generation of an attributed molecular graph, illustrative example for 4-methyl-2-pentanone.

are used and are structured into two phases, the message passing phase and the readout phase, as illustrated in Figure 2.2. In the message passing phase, the molecular structure is learned by employing so-called graph convolutions. Graph convolutions enable each node within the graph to receive information about its neighboring nodes and edges. In the subsequent readout phase, the learned structure information is aggregated into a molecular fingerprint vector which is then mapped to the molecular property of interest by means of, e.g., a multilayer perceptron. All operations within the GNN structure allow for applying backpropagation, enabling end-to-end learning of molecular properties from the molecular graph.

We outline the concept of molecular graphs (Section 2.3.1), the message passing phase (Section 2.3.2), and the readout phase (Section 2.3.3) in more detail, and then describe the end-to-end learning of molecular properties (Section 2.3.4).

2.3.1. Molecular graph

We denote the graph representation of a molecule m with nodes as atoms, also referred to as vertices $v \in V$, and edges as bonds $e_{vw} \in E$ connecting two nodes v and w by $G(m) = \{V, E\}$.

Each node and each edge of the graph G is typically associated with a feature vector that stores atom-/bond-specific information, e.g., the atom type or the bond type. We denote the feature vector of a node v with $\mathbf{f}^v \in F^V$ and the feature vector of an edge e_{vw} with $\mathbf{f}^{e_{vw}} \in F^E$. Typical features are given in Table 5.1 for atoms and in Table 5.2 for bonds. The feature-enriched graph of a molecule m is then denoted by $G(m) = \{V, E, F^V, F^E\}$, often referred to as attributed molecular graph (cf. Figure 2.3).

Table 2.1: Atom features for initial node feature vector [4, 14, 30, 31]. Features are typically encoded as one-hot vectors.

Feature	Description/exemplary values
atom type	type of atom (C, O, S, F)
is in ring	whether the atom is part of a ring
is aromatic	whether the atom is part of an aromatic system
hybridization	sp, sp2, sp3, sp3d, or sp3d2
charge	formal charge of the atom
# bonds	number of bonds the atom is involved in
# Hs	number of bonded hydrogen atoms

Table 2.2: Bond features for initial edge feature vector [4, 14, 30, 31]. Features are typically encoded as one-hot vectors.

Feature	Description/exemplary values
bond type	single, double, triple, or aromatic
conjugated	whether the bond is conjugated
is in ring	whether the bond is part of a ring

Note that the representation of molecules as graphs is a simplification as it omits, e.g., 3D information. Whereas node/edge features can include additional information, e.g., distances or angles between atoms, some characteristics of molecules such as information about stereochemistry or molecule conformation are arguably difficult to capture with a 2D representation such as graphs.

The generation of molecular graphs can be automated by using open-source tools, e.g., RDKit [32], that can also calculate typical node and edge features. Methods from graph-based ML such as GNNs can then be applied to the generated attributed molecular graphs.

2.3.2. MESSAGE PASSING

In the message passing phase, structural information about a molecule is extracted from the molecular graph through graph convolutions. Graph convolutions enable nodes to exchange information with their neighboring nodes, i.e., messages are passed along edges (cf. Figure 2.4). Specifically, graph convolutions combine the feature vector of each node with the features vectors of the corresponding neighboring nodes and edges, i.e., messages containing neighborhood information are passed between nodes (cf. neighborhood aggregation in Section 2.2.2). Stacking a total of L graph convolutional layers together, each

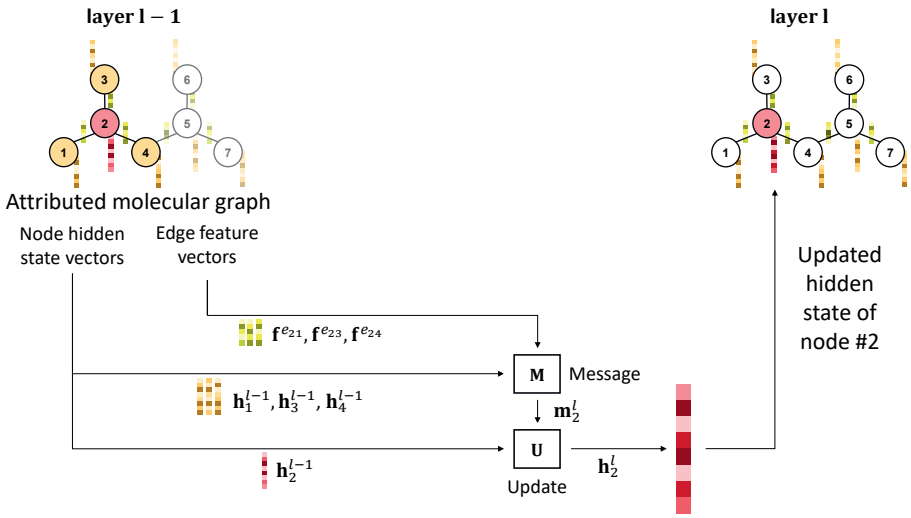


Figure 2.4: Information exchange between nodes within an edge-conditioned graph convolutional layer in the message passing phase of a graph neural network, illustrated update step for node #2, adapted from our publication [14].

node receives information about an environment with a radius of L nodes. The general mathematical form of a graph convolutional (or message passing) layer is described in the following.

In every graph convolutional layer l, each node is represented by a latent or hidden state $\mathbf{h}_v^l$. The hidden state of a node is initialized by its feature vector from the attributed molecular graph, $\mathbf{h}_v^0 = f^v$ with $l = 0$ (cf. Section 2.3.1). The molecular graph then traverses the individual graph convolutional layers $l \in \{1, 2, ..., L\}$. Within a graph convolutional layer l, the hidden states of the nodes within the molecular graph are updated based on the hidden states of the respective neighborhood nodes and the features of the associated edges. The update step can be denoted by the general form of [4, 33]

$$\mathbf{m}_v^l = A_l(\{M_l(\mathbf{h}_v^{l-1}, \mathbf{h}_w^{l-1}, \mathbf{f}^{e_{vw}}) \mid w \in N(v)\}), \tag{2.2}$$

$$\mathbf{h}_v^l = U_l(\mathbf{h}_v^{l-1}, \mathbf{m}_v^l). \tag{2.3}$$

The function M_l denotes the message function mapping the information, the so-called *message*, from one of its neighbor nodes w to node v, with $w \in N(v)$. The message is a function of the preceding hidden states $\mathbf{h}_v^{l-1}$, $\mathbf{h}_w^{l-1}$, and edge e_{vw} with the associated edge feature vector $\mathbf{f}^{e_{vw}}$.[1] The messages from all neighbors of

[1] Note that typically the node states are updated and the edge features remain unchanged. However,

node v are then aggregated in $\mathbf{m}_v^l$ by applying the aggregation function A_l. These neighbor messages together with the preceding hidden state of the node v itself, $\mathbf{h}_v^{l-1}$, are finally combined in the update function U_l to update the hidden state of node v. Consequently, the hidden state of a node in layer l depends on its own previous hidden state, the previous hidden states of its neighboring nodes, and the associated edges. Note that the size of the hidden state vector is a hyperparameter and may vary for different layers.

The L-hop local environment of a node is learned in the message passing phase which builds on neighborhood aggregation (cf. Section 2.2.2). By stacking multiple graph convolutional layers L together, the information of all neighboring nodes within a distance of L nodes is passed to a node, hence the so-called L-hop local environment of each node is learned. Accordingly, the number of layers L is also referred to as the *depth* or *radius* of the network, i.e., the local environment of a node reachable with L so-called hops is extracted. Various approaches exist for the aggregation, the message function M_l, and the update function U_l in graph convolutional layers.

For aggregating the messages passed to a node within a graph convolutional layer by the function A_l, different operators can be used, e.g., mean or max, as well as LSTM aggregators [33]. The sum aggregator has been shown to be more powerful than the mean or max aggregator in terms of distinguishing different neighborhood structures of a node [34]. Thus, typically the sum aggregator is used for molecular property prediction, i.e.,

$$\mathbf{m}_v^l = \sum_{w \in N(v)} M_l(\mathbf{h}_v^{l-1}, \mathbf{h}_w^{l-1}, \mathbf{f}^{e_{vw}}). \tag{2.4}$$

For the message function and the update function in a graph convolutional layer, a wide variety of formulations has been proposed, e.g., graph convolutional network (GCN) [35], graph attention network (GAT) [36], GraphSAGE [33], and higher-order graph convolutions [13]. Further, graph convolutional layers can be combined with a gated recurrent unit (GRU) [4, 37, 38]. Here, we present two graph convolutional approaches: Graph convolutions based only on node features, which we refer to as basic graph convolution, and edge-conditioned graph convolutions, which also take into account edge features. The interested reader is referred to the reviews in [4, 8, 9] for further graph convolutional approaches.

Basic graph convolution. A basic form of graph convolution is a linear transformation of the node states by using a parameter matrix W_1^l to transform the state

there also exists GNN architectures where the edge information changes during the update process or even the whole graph convolutional layer update process is based on messages associated with edges rather than nodes, see, e.g., [31].

of a considered node v and the parameter matrix W_2^l to incorporate messages passed from the neighbors of node v. Specifically, the parameter matrix W_1^l is used to transform the hidden state vector of node v from the previous layer, $\mathbf{h}_v^{l-1}$, whereas the parameter matrix W_2^l is multiplied by the hidden state vectors of the neighboring nodes, $\mathbf{h}_w^{l-1}$. Note that edge features are omitted here. The results of the respective multiplications are summed up and transformed by an activation function σ, e.g., ReLU, resulting in the updated hidden state of the node $\mathbf{h}_v^l$, i.e.,

$$\mathbf{h}_v^l = \sigma(W_1^l \cdot \mathbf{h}_v^{l-1} + \sum_{w \in N(v)} W_2^l \cdot \mathbf{h}_w^{l-1}). \tag{2.5}$$

Edge-conditioned graph convolution. Molecular graphs are typically augmented with edge features, e.g., whether a bond between two atoms is a single, double, or triple bond (cf. Section 2.3.1). To include edge features in the graph convolutions, edge-conditioned graph convolutions have been proposed [39]. The update step in an edge-conditioned convolution is illustrated in Figure 2.4. For edge-conditioned graph convolutions the message function in Equation (2.2) is defined by

$$M_l = \text{ANN}_{W_E^l}(\mathbf{f}^{e_{vw}}) \cdot \mathbf{h}_w^{l-1} \quad \text{with} \quad w \in N(v), \tag{2.6}$$

where ANN denotes an artificial neural network mapping the edge features to a parameter matrix that is multiplied with the hidden states of the neighbors of node v, $\text{ANN}_{W_E^l} : \mathbf{f}^{e_{vw}} \mapsto W_E^l$. The hidden state of the node v is again multiplied by a parameter matrix W_V^l, similarly to the standard graph convolution. The edge-conditioned message function combined with the sum aggregator then results in the following update process:

$$\mathbf{h}_v^l = \sigma(W_V^l \cdot \mathbf{h}_v^{l-1} + \sum_{w \in N(v)} \text{ANN}_{W_E^l}(\mathbf{f}^{e_{vw}}) \cdot \mathbf{h}_w^{l-1}) \tag{2.7}$$

2.3.3. READOUT

In the *readout phase*, the structure information learned during message passing and stored in the hidden nodes is first aggregated into a molecular fingerprint vector. We denote the aggregation by the pooling function P, i.e.,

$$\mathbf{h}_{FP} = P(\{\mathbf{h}_v^L \mid v \in V\}), \tag{2.8}$$

that combines the hidden states of the nodes of the last convolutional layer L into the molecular fingerprint $\mathbf{h}_{FP}$ (cf. Figure 2.5), a continuous molecular vector representation. The size of the molecular fingerprint is implicitly defined by the size of the node hidden state vectors in the last graph convolutional layer which is a hyperparameter (cf. Section 2.3.2).

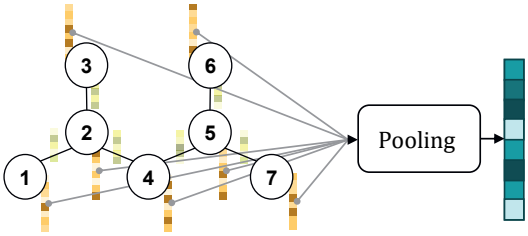


Figure 2.5: Pooling step in a graph neural network, illustrative example for 4-methyl-2-pentanone.

Different pooling functions exist and range from simple mathematical operations, e.g., mean, sum, or max, to advanced neural network approaches [8]. A crucial condition for these pooling functions is that the function should be invariant to the order of nodes, hence permutation/isomorphism invariant [9, 34], and size-agnostic, i.e., applicable to a varying number of nodes. Flexible pooling functions have been developed in recent years [40–42], one popular example being the set2set approach [40]. The set2set model is designed to operate on input sets with an invariance for the order of the objects [4, 40] and employs an attention mechanism including a LSTM [40] that yields the molecular fingerprint after T steps, i.e., $\mathbf{h}_{FP} = \mathbf{q}^*_{t=T}$, by executing the iterative scheme

$$\begin{aligned} \mathbf{q}_t &= \mathrm{LSTM}\left(\mathbf{q}^*_{t-1}\right), \\ \mathbf{a}_{v,t} &= \mathrm{softmax}\left(\mathbf{h}^L_v \cdot \mathbf{q}_t\right), \\ \mathbf{r}_t &= \sum_v \mathbf{a}_{v,t} \cdot \mathbf{h}^L_v, \\ \mathbf{q}^*_t &= \mathbf{q}_t \,\|\, \mathbf{r}_t, \end{aligned} \tag{2.9}$$

where $\mathbf{q}_t$ is a query vector for iteration t providing information from the last iteration by an LSTM, $\mathbf{a}_{v,t}$ is an attention vector yielding the attention of a node v by applying the softmax function, $\mathbf{r}_t$ is the attention readout vector calculated by the attention-weighted sum of the single node hidden states, and $\mathbf{q}_t$ is a concatenation ($\|$) of the query vector and the attention readout vector. The vector $\mathbf{q}^*_{t-1}$ can be initialized with the zero-vector, i.e., $\mathbf{q}^*_{-1} = \mathbf{0}$.

Finally, the molecular fingerprint $\mathbf{h}_{FP}$ must be mapped to the property of interest. Here, standard regression methods can be employed. In GNNs, typically, a feedforward neural network, a multilayer perceptron (MLP), is used to compute the molecular property $\hat{p}$ as a function of the molecular fingerprint vector, i.e.,

$$\hat{p} = \mathrm{MLP}(\mathbf{h}_{FP}). \tag{2.10}$$

2.3.4. END-TO-END LEARNING

We denote the entire feedforward function f_{GNN} of the GNN for prediction of a molecular property $\hat{p}$ from the molecular graph $G(m)$ by

$$f_{GNN} : G(m) \mapsto \hat{p}. \tag{2.11}$$

All mathematical operators used in the GNN for the mapping f_{GNN} can be integrated into the backpropagation scheme of ML models. In particular, backpropagation can be used to learn the parameters of all three functions in the message passing phase (cf. Section 2.3.2), i.e., the message function M_l, the aggregation function, and the update function U_l. Likewise, the pooling function and the feedforward ANN in the readout phase (cf. Section 2.3.3) allow for backpropagation. Thus, the whole GNN structure can be trained with backpropagation in a supervised setup [4, 33]. Therefore, GNNs are able to learn in an end-to-end manner, from the molecular graph to the property of interest. As the molecular fingerprint automatically adapts during the GNN training to capture the structural information that is important for the prediction task at hand [43], GNNs circumvent the definition and selection of informative molecular descriptors as it is required in QSPR/QSAR modeling.

2.4. NUMERICAL EXAMPLES

To demonstrate typical workflows of applying GNNs for molecular property prediction, we consider two examples and focus on data pre-processing, model setup and training, and result analysis. Both examples are implemented in Python using the open-source extensions PyG (PyTorch Geometric) [44], a package for geometric ML which includes a wide range of GNN architectures, and RDKit [32], a chemoinformatics package (cf. Section 2.3.1). The first example is a regression task, i.e., a real-valued property of a molecule, herein the boiling point, shall be predicted. The second example is a classification problem. Specifically, a molecule shall be categorized as biodegradable or non-biodegradable. The examples show that GNN models with standard architecture can provide high prediction accuracies, enabling fast and accurate predictions that can, for instance, be used in computer-aided molecular design.

2.4.1. REGRESSION EXAMPLE: BOILING POINT PREDICTION

The normal boiling point is often used to assess the general suitability of a chemical component as a raw/working material, a reactant, or a product within chemical process design. For novel compounds, experimentally determined boiling points are often not readily available. In this example, we demonstrate the development of a GNN for normal boiling point prediction.

Data set and preprocessing

We use the normal boiling point data set provided by Alsheri et al. [45] which contains 5,276 boiling points for 5,089 molecules of various classes, i.e., pure hydrocarbons, hydrocarbons with additional atoms such as oxygen, nitrogen, or fluorine, as well as multi-functional molecules. Initially, we randomly select 10 % of the data set for testing, i.e., 509 molecules are set aside. Note that the test set is kept unchanged in the following. The remaining 4,749 data points are split randomly into 90 % training set and 10 % validation set. We convert the molecules provided as SMILES strings to attributed molecular graphs with the atom and bond features provided in Table 5.1 and Table 5.2, respectively, with the help of RDKit [32]. Note that we extend the atom features stated in Table 5.1 to the atom types occurring in the data set (C, O, N, F, S, Cl, P, I, Br, Si) and exclude atom features that do not occur or do not vary, i.e., sp3d and atom charge, resulting in 20 atom features and 6 bond features in total. We standardize the boiling point values to a Gaussian with a zero mean and standard deviation of one (Z-score normalization) for training purposes.

Model setup and training

We use a GNN architecture with three edge-conditioned graph convolutional layers in the message passing phase and sum pooling followed by a three-layer MLP in the readout phase, cf. Section 2.3. For the graph convolutions, the hidden state vectors are set to a dimension of 64 yielding a 64-dimensional molecular fingerprint. The edge-MLPs within the graph convolutions have three layers with 6, 128, and 4092 (=64^2) neurons, respectively. The three layers in the MLP for the readout have 64, 32, and 1 neurons, with the last neuron yielding the boiling point prediction.

To train the GNN model, we minimize the loss, i.e., the mean squared error between boiling point predictions and labels, by using the Adam optimizer. We train the model for a maximum of 300 epochs and apply early stopping with a patience of 25 epochs, i.e., the training is stopped early if the validation mean absolute error does not decrease for 25 consecutive epochs. The learning rate is initialized with a value of 0.001 and decayed with a factor of 0.8 if the validation error does not decrease for 3 consecutive epochs (patience). We further use a batch size of 8. In addition to training the single GNN prediction model, we employ ensemble learning which means that we train 40 GNN models with different random splits of the data set into training and validation sets and average the predictions of the 40 GNNs to compute the final boiling point prediction.

Prediction results

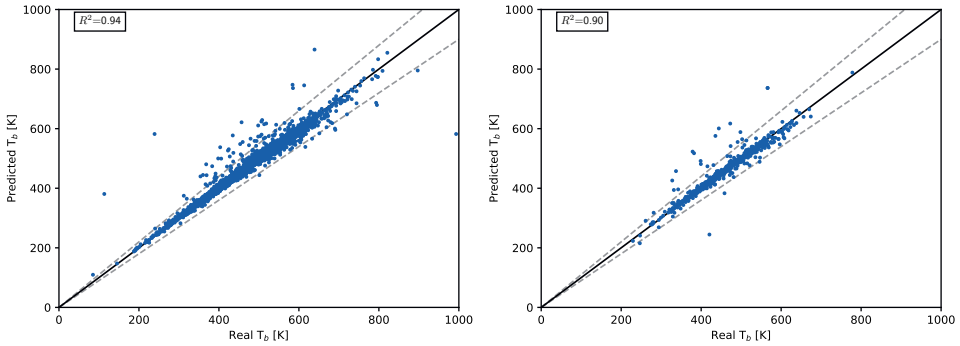
Table 2.3 shows the boiling point T_b prediction accuracies achieved by the GNN. Note that we report the average accuracy and corresponding standard deviation with respect to training, validation, and test set data. As the ensemble GNN contains 40 models with varying training and validation splits, we state a single aggregated training and validation set accuracy. For the single GNN model, the statistics are computed over 40 single models.

The mean absolute error (MAE) of the single GNN models on the test set is 14.4 K which corresponds to a mean absolute percentage error (MAPE) of about 3.2 % and a coefficient of determination R^2 of 0.86. Comparing the prediction quality of the single GNN models to the ensemble GNN, we observe a clear advantage for the ensemble, i.e., an increase in accuracy with respect to both the combined training/validation set and the test set. Specifically, the MAE on the test set is reduced by 2.5 K to 11.9 K, resulting in an R^2 of 0.90. The increased accuracy results from over- and underpredictions of single GNNs for the same molecule being partly balanced out by taking the average prediction.

Model setup	Boiling point, T_b			
	MAE/K	R^2	MAPE/%	MAXPE/%
single GNN (training set)	8.7 ± 2.1	0.93 ± 0.04	1.9 ± 0.4	48.4 ± 13.2
single GNN (validation set)	13.3 ± 1.8	0.85 ± 0.13	2.9 ± 0.3	33.7 ± 23.8
single GNN (test set)	14.4 ± 1.3	0.86 ± 0.04	3.2 ± 0.3	41.9 ± 4.4
ensemble of 40 GNNs (training/validation set)	7.1	0.94	1.5	42.5
ensemble of 40 GNNs (test set)	11.9	0.90	2.6	41.9

Table 2.3: Model accuracies for boiling point (T_b) prediction: mean absolute error (MAE) in Kelvin (K), coefficient of determination (R^2), mean absolute percentage error (MAPE), and maximum absolute percentage error (MAXPE). For the single GNN model, the statistics are computed over 40 single models.

Experimental boiling points and predictions by the ensemble for the molecules within the training/validation set and the test set are visualized in the parity plots in Figure 2.6. Most predictions deviate less than ± 10 % from the actual values, indicated by the grey dashed lines (Figure 2.6). More specifically, for the training/validation set about 99 % and for the test set about 95 % of the data points are predicted with an absolute error below 10 %. For both sets some outliers can be observed. Nevertheless, the total number of outliers is low and the residuals are approximately normally distributed. We thus find an overall high prediction quality.



(a) Training/validation set

(b) Test set

Figure 2.6: Parity plots for GNN ensemble. Grey dashed lines indicate ± 10 % error. Note that one sample within the training/validation set with a measured boiling point of about 2500 K is not shown for better visualization.

2.4.2. CLASSIFICATION EXAMPLE: BIODEGRADABILITY PREDICTION

The assessment of properties that indicate environmental impacts and hazards of chemicals is a critical step in the design of sustainable products and processes. A property of paramount importance here is the persistence of chemicals in the environment, as the accumulation of persistent materials may exhibit long-term toxic effects. This classification example aims at categorizing molecules into readily biodegradable or non-readily biodegradable substances by means of GNN modeling.

Data set and preprocessing

For binary biodegradability classification, Mansouri et al. [46] provided the "UCI QSAR Biodegradation Data-Set", a data set consisting of 1,725 molecules with a binary biodegradability label. The class of readily biodegradable compounds contains 547 molecules, whereas 1,178 molecules are considered non-readily biodegradable. We use the pre-defined test set with 670 molecules from Mansouri et al. [46] and the remaining 1,055 data points with a random split into 90 % training and 10 % validation data for training purposes. Again, attributed molecular graphs with atom and bond features as provided in Table 5.1 and Table 5.2, respectively, are inputted to the GNN model, where we extend the atom features to the atom types occurring in the data set (C, O, N, F, Si, S, P, Cl, Ti, Cu, Br, Sn, I, Bi) and exclude one atom feature not occurring in the data, i.e., sp3d, yielding 27 atom features and 6 bond features in total.

Model setup and training

The GNN model architecture is similar to that of the regression example described above. We employ three edge-conditioned graph convolutional layers in the message passing phase and apply sum pooling with a three-layer MLP for the readout. The hidden state vectors of the graph convolutions and molecular fingerprint have 64 dimensions; the edge-MLPs have three layers with, respectively, 6, 128, and 4092 (=64^2) neurons. The three layers in the readout-MLP have 64, 32, and 1 neurons. In case of binary classification, the readout-MLP predicts the probability of a positive class, herein, readily biodegradable, by applying a sigmoid function in the last layer. We classify a molecule as readily biodegradable if the probability is higher than a threshold value of 0.5, a hyperparameter of the model.

To train the GNN model, we use the Adam optimizer with the binary cross-entropy loss function, a batch size of 8, an initial learning rate of 0.001, and learning rate decay with a factor of 0.8 and a patience of 3 epochs. The training ends either after 300 epochs or when the validation cross-entropy loss does not decrease for 25 consecutive epochs (early stopping). We again perform single GNN modeling and ensemble learning. In the latter case, we train 40 GNN models with different random data set splits and average the property predictions, i.e., a molecule is classified as biodegradable if the average probability prediction of the 40 models yields a value higher than 0.5.

Prediction results

Table 2.4 shows several performance indicators of the GNN models. The average performance and corresponding standard deviation for 40 single GNN models with respect to training, validation, and test sets are reported. For the GNN ensemble, the performance on the aggregated training/validation set (cf. Section 2.4.1) and the test set is indicated.

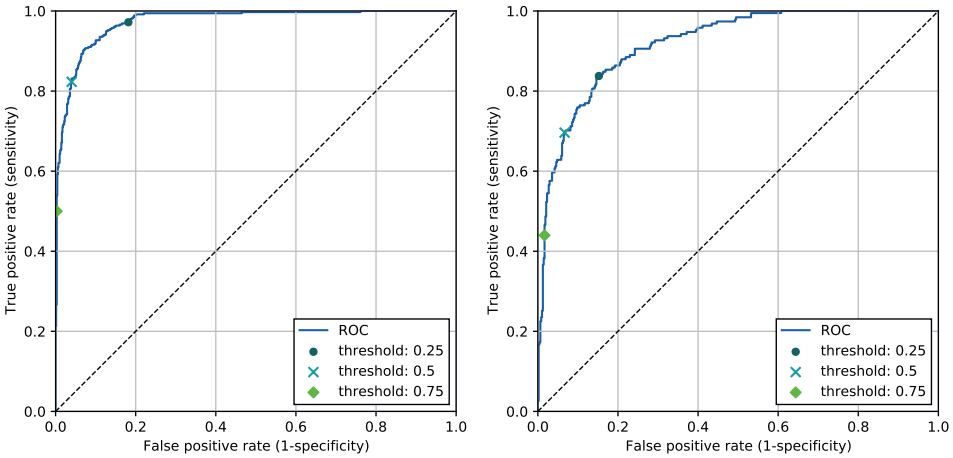
Model setup	Biodegradation			
	ACC	SP	SN	AUROC
single GNN (training set)	0.89 ± 0.03	0.94 ± 0.03	0.77 ± 0.10	0.96 ± 0.02
single GNN (validation set)	0.86 ± 0.04	0.93 ± 0.03	0.72 ± 0.09	0.93 ± 0.03
single GNN (test set)	0.85 ± 0.01	0.93 ± 0.03	0.66 ± 0.07	0.91 ± 0.01
ensemble of 40 GNNs (training/validation set)	0.91	0.96	0.82	0.97
ensemble of 40 GNNs (test set)	0.87	0.93	0.70	0.92

Table 2.4: Model accuracies for ready biodegradation prediction: Accuracy is provided by accuracy score (ACC), specificity (SP), sensitivity (SN), all at a threshold value of 0.5, and area under the receiver operating characteristic curve (AUROC). For the single GNN model, the statistics are computed over 40 single models.

The averaged accuracy of the single GNN models amounts to 0.89 for the

training set, 0.86 for the validation set, and 0.85 for the test set, indicating a high prediction quality, an observation that is further emphasized by specificity values exceeding 0.9 in case of all sets. The sensitivity values are, however, visibly lower, e.g., a sensitivity of 0.66 is found for the test set. The low sensitivity and high specificity point to the model's tendency of being cautious in classifying a molecule as readily biodegradable. Note that altering the threshold value on the predicted probability of a molecule being classified as readily biodegradable can be used to balance specificity and sensitivity. We illustrate this trade-off with the receiver operating characteristic (ROC) curve below.

We observe a slightly better prediction quality for the ensemble GNN. For example, both the accuracy and sensitivity scores increase by 0.02 and 0.03, respectively. The specificity with a value of 0.93, however, remains unchanged. Overall, the ensemble thus only slightly enhances prediction performance in this example.



(a) Training/validation set

(b) Test set

Figure 2.7: Receiver operating characteristic (ROC) curve in blue for GNN model ensembling.

Figure 2.7 shows the ROC curve for the training/validation set and the test set when using the GNN ensemble. The ROC illustrates the true positive rate (sensitivity) versus the false positive rate (1−specificity) when varying the threshold probability value for deciding if a molecule is classified as readily biodegradable. For example, if the threshold value is lowered, more molecules will be classified as biodegradable which increases the true positive rate but also the false positive rate. A ROC curve with many high true positive and low false positive rate points, i.e., a curve located in the upper left corner of the graph, indicates a balanced classification model. In Figure 2.7, we observe that the ROC curves for

training/validation sets and test set are approaching that upper left region. The good model performance is also reflected in the area under the receiver operating characteristic curve (AUROC) values (cf. Table 2.4). All values being larger than 0.9 is further evidence of an overall high prediction quality. In fact, the GNN models for classifying molecules into readily biodegradable or not readily biodegradable yield similar prediction accuracy than state-of-the-art QSPR models [46–48].

2.5. CONCLUDING REMARKS

Graph neural networks (GNNs) have emerged as a powerful method for predicting physico-chemical properties of molecules allowing end-to-end learning from the molecular graph to the property of interest. A principal advantage of GNNs is their straightforward applicability in situations where informative molecular descriptors relevant to the property of interest are not known and where the application of classical QSAR/QSPR modeling is therefore difficult. We have illustrated the application of GNNs with standard architectures to property regression and classification tasks, i.e., the prediction of the normal boiling point and the prediction of biodegradability, respectively. Providing fast and high-quality molecular property predictions, GNNs provide a promising tool for in silico screening of molecules as well as computer-aided molecular design.

GNNs constitute a fast moving research field, with novel GNN architectures being developed. The novel architectures differ in many ways, for example, whether only representations of nodes [4, 33], edges [31], or both [29] are learned and what type of graph convolution and pooling function is applied. Recent GNNs for molecular property prediction also incorporate physical knowledge, i.e., the message passing scheme is integrated with physical knowledge, e.g., SchNet [49], DimeNet [50], MXMNet [6]. This includes the incorporation of directional information, such as interatomic distances and angles between atom pairs, into the message function $M_l(\cdot)$. Additional GNN extensions include applying a global message passing that enables direct interatomic information exchange between atoms that are not connected by a chemical bond, in contrast to local message passing as described above [6]. Furthermore, GNNs have become more expressive from a graph theoretical perspective, i.e., more powerful in distinguishing graphs. It was shown that GNNs are as powerful as the one-dimensional Weisfeiler-Lehmann algorithm in discriminating graphs [13, 34]. By including higher-order graph features [13] or regular cell complexes [51], the expressiveness of GNNs can be increased. For further insights on the theoretical foundations, we refer the interested reader to review articles [13, 34, 52].

Future GNN research will also need to focus on uncertainty quantification and interpretability of GNN predictions. Next to the methodological advances, further application areas of GNNs in molecular property prediction will be explored, as

well as beyond. In chemical engineering, for instance, also process flow sheets or reaction networks can be represented as graphs, suggesting another huge potential for GNN applications [53].

2

REFERENCES

[1] L. P. Hammett, *Some relations between reaction rates and equilibrium constants.* Chemical Reviews **17**, 125 (1935).

[2] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning,* Nature **521**, 436 (2015).

[3] D. C. Elton, Z. Boukouvalas, M. D. Fuge, and P. W. Chung, *Deep learning for molecular design - A review of the state of the art,* Molecular Systems Design and Engineering **4**, 828 (2019), arXiv:1903.04388 .

[4] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, *Neural message passing for quantum chemistry,* 34th International Conference on Machine Learning, ICML 2017 **3**, 2053 (2017), arXiv:1704.01212 .

[5] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande, *Moleculenet: a benchmark for molecular machine learning,* Chemical science **9**, 513 (2018).

[6] S. Zhang, Y. Liu, and L. Xie, *Molecular mechanics-driven graph neural network with multiplex graph for molecular structures,* arXiv preprint arXiv:2011.07457 (2020).

[7] P. Pope, S. Kolouri, M. Rostrami, C. Martin, and H. Hoffmann, *Discovering Molecular Functional Groups Using Graph Convolutional Neural Networks,* (2018), arXiv:1812.00265 .

[8] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, *A comprehensive survey on graph neural networks,* IEEE Transactions on Neural Networks and Learning Systems **32**, 4 (2021).

[9] Z. Zhang, P. Cui, and W. Zhu, *Deep learning on graphs: A survey,* IEEE Transactions on Knowledge and Data Engineering **34**, 249 (2022).

[10] T. Sterling and J. J. Irwin, *ZINC 15 - Ligand Discovery for Everyone,* Journal of Chemical Information and Modeling **55**, 2324 (2015).

[11] L. Ruddigkeit, R. van Deursen, L. C. Blum, and J.-L. Reymond, *Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17,* Journal of Chemical Information and Modeling **52**, 2864 (2012).

[12] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld, *Quantum chemistry structures and properties of 134 kilo molecules,* Scientific Data **1**, 140022 (2014).

[13] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, *Weisfeiler and leman go neural: Higher-order graph neural networks,* 33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019, and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019 , 4602 (2019), arXiv:1810.02244 .

[14] A. M. Schweidtmann, J. G. Rittig, A. König, M. Grohe, A. Mitsos, and M. Dahmen, *Graph Neural Networks for Prediction of Fuel Ignition Quality,* Energy & Fuels **34**, 11395 (2020).

[15] E. I. Sanchez Medina, S. Linke, M. Stoll, and K. Sundmacher, *Graph neural networks for the prediction of infinite dilution activity coefficients,* Digital Discovery (2022), 10.1039/D1DD00037C.

[16] A. R. Katritzky, M. Kuanar, S. Slavov, C. D. Hall, M. Karelson, I. Kahn, and D. A. Dobchev, *Quantitative correlation of physical and chemical properties with chemical structure: utility for prediction,* Chemical Reviews **110**, 5714 (2010).

[17] A. Cherkasov, E. N. Muratov, D. Fourches, A. Varnek, I. I. Baskin, M. Cronin, J. Dearden, P. Gramatica, Y. C. Martin, R. Todeschini, V. Consonni, V. E. Kuz'min, R. Cramer, R. Benigni, C. Yang, J. Rathman, L. Terfloth, J. Gasteiger, A. Richard, and A. Tropsha, *QSAR modeling: where have you been? Where are you going to?* Journal of Medicinal Chemistry **57**, 4977 (2014).

[18] R. Todeschini and V. Consonni, *Molecular descriptors for chemoinformatics*, 2nd ed., Methods and Principles in Medicinal Chemistry, Vol. 41 (Wiley-VCH, Weinheim, Germany, 2009).

[19] R. Todeschini and V. Consonni, *Handbook of Molecular Descriptors* (Wiley-VCH, Weinheim, Germany, 2000).

[20] S. W. Benson, F. R. Cruickshank, D. M. Golden, G. R. Haugen, H. E. O'Neal, A. S. Rodgers, R. Shaw, and R. Walsh, *Additivity rules for the estimation of thermochemical properties,* Chemical Reviews **69**, 279 (1969).

[21] K. G. Joback and R. C. Reid, *Estimation of pure-component properties from group-contributions,* Chemical Engineering Communications **57**, 233 (1987).

[22] H. L. Morgan, *The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service,* Journal of Chemical Documentation **5**, 107 (1965).

[23] R. C. Glem, A. Bender, C. H. Arnby, L. Carlsson, S. Boyer, and J. Smith, *Circular fingerprints: flexible molecular descriptors with applications from physical chemistry to adme,* IDrugs: the Investigational Drugs Journal **9**, 199 (2006).

[24] D. Rogers and M. Hahn, *Extended-connectivity fingerprints,* Journal of Chemical Information and Modeling **50**, 742 (2010).

[25] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, *Convolutional networks on graphs for learning molecular fingerprints,* in *Advances in Neural Information Processing Systems 28* (Curran Associates, Inc., 2015) pp. 2224–2232.

[26] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, *Geometric deep learning: going beyond euclidean data,* IEEE Signal Processing Magazine **34**, 18 (2017).

[27] M. Gori, G. Monfardini, and F. Scarselli, *A new model for learning in graph domains,* in *Proceedings of the IEEE International Joint Conference on Neural Networks, 2005* (IEEE, Montreal, QC, Canada, 2005) pp. 729–734.

[28] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, *The graph neural network model,* IEEE Transactions on Neural Networks **20**, 61 (2009).

[29] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, *Molecular graph convolutions: moving beyond fingerprints,* Journal of Computer-Aided Molecular Design **30**, 595 (2016).

[30] C. W. Coley, R. Barzilay, W. H. Green, T. S. Jaakkola, and K. F. Jensen, *Convolutional embedding of attributed molecular graphs for physical property prediction,* Journal of Chemical Information and Modeling **57**, 1757 (2017).

[31] K. Yang, K. Swanson, W. Jin, C. Coley, P. Eiden, H. Gao, A. Guzman-Perez, T. Hopper, B. Kelley, M. Mathea, A. Palmer, V. Settels, T. Jaakkola, K. Jensen, and R. Barzilay, *Analyzing learned molecular representations for property prediction,* Journal of Chemical Information and Modeling **59**, 3370 (2019).

[32] Greg Landrum, *RDKit: Open-source cheminformatics software,* Accessed on 01.04.2022.

[33] W. Hamilton, Z. Ying, and J. Leskovec, *Inductive representation learning on large graphs,* in *Advances in Neural Information Processing Systems 30 (NIPS 2017)* (2017) pp. 1024–1034.

[34] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, *How powerful are graph neural networks?* arXiv preprint arXiv:1810.00826v3 (2018).

[35] T. N. Kipf and M. Welling, *Variational graph auto-encoders,* arXiv preprint arXiv:1611.07308 (2016), arXiv:1611.07308 .

[36] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, *Graph attention networks,* 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings (2018).

[37] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, *Learning phrase representations using rnn encoder-decoder for statistical machine translation,* arXiv preprint arXiv:1406.1078v3 (2014).

[38] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, *Gated graph sequence neural networks,* arXiv preprint arXiv:1511.05493v4 (2015).

[39] M. Simonovsky and N. Komodakis, *Dynamic edge-conditioned filters in convolutional neural networks on graphs,* arXiv preprint arXiv:1704.02901v3 (2017).

[40] O. Vinyals, S. Bengio, and M. Kudlur, *Order matters: Sequence to sequence for sets,* 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings , 1 (2016), arXiv:1511.06391 .

[41] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, *An end-to-end deep learning architecture for graph classification,* Proceedings of the AAAI Conference on Artificial Intelligence **32** (2018).

[42] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, *Hierarchical graph representation learning with differentiable pooling,* in *Advances in Neural Information Processing Systems* (2018) pp. 4800–4810.

[43] C. W. Coley, R. Barzilay, W. H. Green, T. S. Jaakkola, and K. F. Jensen, *Convolutional Embedding of Attributed Molecular Graphs for Physical Property Prediction,* Journal of Chemical Information and Modeling **57**, 1757 (2017).

[44] M. Fey and J. E. Lenssen, *Fast graph representation learning with pytorch geometric,* arXiv preprint arXiv:1903.02428v3 (2019).

[45] A. S. Alshehri, A. K. Tula, F. You, and R. Gani, *Next generation pure component property estimation models: With and without machine learning techniques,* AIChE Journal **49**, 11 (2021).

[46] K. Mansouri, T. Ringsted, D. Ballabio, R. Todeschini, and V. Consonni, *Quantitative structure-activity relationship models for ready biodegradability of chemicals,* Journal of Chemical Information and Modeling **53**, 867 (2013).

[47] G. B. Goh, K. Sakloth, C. Siegel, A. Vishnu, and J. Pfaendtner, *Multimodal deep neural networks using both engineered and learned representations for biodegradability prediction,* arXiv preprint arXiv:1808.04456v2 (2018).

[48] M. Lee and K. Min, *A comparative study of the performance for predicting biodegradability classification: The quantitative structure-activity relationship model vs the graph convolutional network,* ACS Omega **7**, 3649 (2022).

[49] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller, *SchNet - A deep learning architecture for molecules and materials,* Journal of Chemical Physics **148**, 1 (2018), arXiv:1712.06113 .

[50] J. Klicpera, J. Groß, and S. Günnemann, *Directional message passing for molecular graphs,* arXiv preprint arXiv:2003.03123 (2020).

[51] C. Bodnar, F. Frasca, N. Otter, Y. G. Wang, P. Liò, G. F. Montufar, and M. Bronstein, *Weisfeiler and lehman go cellular: CW networks,* in *Advances in Neural Information Processing Systems*, Vol. 34 (Curran Associates, Inc., 2021) pp. 2625–2640.

[52] C. Morris, Y. Lipman, H. Maron, B. Rieck, N. M. Kriege, M. Grohe, M. Fey, and K. Borgwardt, *Weisfeiler and leman go machine learning: The story so far,* arXiv preprint arXiv:2112.09992 (2021).

[53] A. M. Schweidtmann, E. Esche, A. Fischer, M. Kloft, J.-U. Repke, S. Sager, and A. Mitsos, *Machine learning in chemical engineering: A perspective,* Chemie Ingenieur Technik **93**, 2029 (2021).

3

DEEP REINFORCEMENT LEARNING FOR PROCESS DESIGN: REVIEW AND PERSPECTIVE

The transformation towards renewable energy and feedstock supply in the chemical industry requires new conceptual process design approaches. Recently, deep reinforcement learning, a subclass of machine learning, has shown the potential to solve complex decision-making problems and aid sustainable process design. However, its suitability in static process design still needs to be examined. We discuss the advantages and disadvantages of reinforcement learning for process design. Then, we survey state-of-the-art research through three major elements: (i) information representation, (ii) agent architecture, and (iii) environment and reward. Moreover, we discuss perspectives on underlying challenges and promising future works to unfold the full potential of reinforcement learning for process design in chemical engineering.

This chapter has been published as "Deep reinforcement learning for process design: Review and perspective" by Gao, Q., & Schweidtmann, A. M., Current Opinion in Chemical Engineering **44**, 101012 (2024)

3.1. INTRODUCTION

The chemical industry is facing a rapid paradigm shift towards a circular economy based on renewable energy and feedstock supply [1, 2]. This poses several challenges for conceptual process design due to the increasing complexity of the design task, the lack of experienced engineers, and the pressure on improving sustainability and profitability while shortening development times. Thus, there is a need for new methodologies and tools that support engineers to design sustainable processes in a more efficient way.

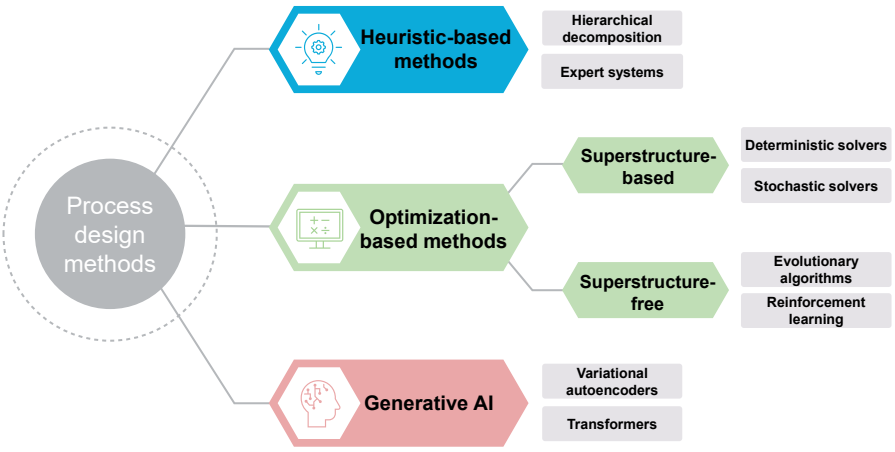


Figure 3.1: Overview of computer-aided process design methods.

Computer-aid process design (CAPD) is widely used in process systems engineering (PSE) for conceptual process design [3, 4], which can be classified into three methodologies as illustrated in Figure 3.1: (i) heuristic-based methods; (ii) optimization-based methods; and (iii) the emerging field of generative artificial intelligence (AI). Heuristic-based methods rely on a set of rules derived from experience, insights, and engineering knowledge, making them the most commonly used approaches due to their ease of application. Optimization-based approaches are commonly used to identify optimal design and operating variables for given process structures. Particularly, derivative-based optimization algorithms are already available in commercial process simulation software to determine those. To determine optimal process structures, superstructure-based methods are the de facto state of the art, where possible process alternatives are modeled and subsequently solved by mixed-integer nonlinear optimization (MINLP) methods [5, 6]. Within this paradigm, two solver types are typically uti-

lized: deterministic (e.g., branch and bound algorithms like BARON or MAiNGO), and stochastic solvers (e.g., genetic algorithms). While superstructure methods have been very successful in PSE, they also have many shortcomings that limit industrial applications [7], including (manual) setup of all process alternatives, the need to implement process models in an optimization environment, and the difficulties of solving resulting MINLPs. Superstructure-free methods dispense the need for predefined superstructures. For example, evolutionary algorithms, typically adopt a two-level decomposition approach [6]. First, they generate alternative flowsheets which are then evaluated through an optimization algorithm. Finally, a few very recent works proposed the use of generative AI methods for the generation of process structures [8, 9]. However, those works require large training data and are not the focus of this review.

Recently, deep reinforcement learning (RL) has shown its potential to solve complex sequential dynamic decision-making problems at human-like or even superhuman level [10–12]. RL is a computational approach for goal-directed learning and decision-making through the direct interaction of an agent with its environment [13]. RL is primarily developed to solve discrete-time dynamic optimization problems formulated as Markov Decision Processes. Consequently, RL is based on the Bellman optimality equation, which is similar to the Hamilton-Jacobi-Bellman (HJB) equation and Pontryagin's Maximum Principle (PMP) for continuous state and action spaces in the control theory. Also, RL has seen its first applications in chemical engineering for process control [14, 15] and scheduling [16, 17]. Notably, Yokogawa is even using RL to operate an industrial chemical process since 2022.

This review and perspective paper aims to provide a critical examination of the application of RL for process design. In the context of process design, RL can be considered a superstructure-free and model-free method, which iteratively places unit operations with corresponding design and operating variables. It evaluates the resulting flowsheets at every iteration and aims to maximize the objective functions. In the recent literature, there have been a few first steps towards applying RL as a static optimization algorithm for stationary process design including absorption–stripping process [18], energy systems design [19, 20], unit operation design [21], separation process [22–29], solvent extraction process design [30], single mixed-refrigerant process design [31], and synthesis reaction process design [32–34].

The use of RL for stationary process design is controversial in scientific discussions. Most notably, RL is better suited for dynamic, sequential-decision problems rather than static ones. Also, clear comparisons and benchmarks between RL and other design methods are lacking in previous literature. Therefore, at the moment, the suitability of RL for process design is questionable and remains an

open question. Here, we present the main differences between RL and existing superstructure-free methods in the context of process design:

- **Computational efficiency (static vs. dynamic optimization)** - RL is better suited for dynamic, sequential-decision problems rather than static ones. However, prior studies have applied RL to stationary process design, constituting static optimization problems. This may drastically increase complexity and decrease computational efficiency (called "sample efficiency" in RL). As a side note, the capability of RL in dynamic optimization paves the way for solving integrated design and operation problems (see Section 3.3.5).

- **Iterative build-up vs. full flowsheet generation** - RL sequentially generates flowsheets (unit by unit), differing from evolutionary superstructure-free methods that construct flowsheets as a whole. While this sequential build-up increases computational complexity, there are also potential advantages. Generating feasible flowsheets and simulating them is challenging, often causing convergence problems in evolutionary superstructure-free methods. In contrast, the iterative strategy of RL promotes convergence and intermediate simulations of incomplete flowsheets may provide valuable information for learning. However, this additional information comes at the cost of additional simulation time.

- **Inference (online) vs. optimization (offline)** - A significant distinction between RL and other optimization methods lies in the solution time for similar, recurring problems. RL involves training a policy once, which is then utilized during inference to rapidly predict near-optimal solutions, offering a significant advantage in time-sensitive control applications. In contrast, classical optimization algorithms solve problems individually, typically requiring long runtimes for each problem instance. In process design, long optimization times are usually not an issue (unless it becomes intractable in the case of large non-convex MINLPs). Thus, classical optimization methods are usually well-suited. However, the rapid inference capability of RL may also provide new opportunities for process design. For example, RL agents might be integrated into flowsheet simulation software to automatically and immediately suggest near-optimal design options to users. Also, fast solutions of design problems can be advantageous when a large numbers of design problems need to be solved (e.g., as subproblems in larger optimization studies).

- **Learning capacity** - RL possesses a substantially greater learning capacity compared to standard evolutionary methods (e.g., more trainable parameters). For instance, state-of-the-art deep RL algorithms can incorporate

extensive networks of learnable parameters, potentially exceeding billions of parameters. This enhanced learning potential facilitates inference and allows for retaining information, unlike genetic algorithms which typically lose details about previous populations. This substantial learning capacity of RL holds the potential for learning more complex dependencies between design actions and results. However, the high learning capacity also presents severe drawbacks such as a vast amount of training data and extensive training duration (measured in epochs within RL).

When comparing RL with standard optimization methods, its most notable advantages include larger learning capacity and inference ability. However, at the same time, RL typically demands significantly more training simulations than static optimization solvers, which leaves doubt on whether its potential advantages outweigh the disadvantages in the context of process design. Current literature applying RL to process design neglects its inference capabilities, learning capacity, and dynamic optimization capabilities, predominantly utilizing the training phase of RL as an evolutionary optimization strategy for static problems. Thereby, they essentially merge the drawbacks of both worlds. Additionally, there is a lack of computational comparison between RL and traditional process design methods. In the following, we critically examine the existing literature on RL in process design (Section 3.2) and highlight future perspectives (Section 3.3).

3.2. STATE OF THE ART

The general framework of RL in process design is shown in Figure 3.2. The agent learns to design processes by iteratively placing unit operations with design and operating variables, and simulating the resulting processes in the environment, ultimately obtaining the optimal policy π^* which designs optimal processes. Mathematically, this problem can be formulated as Markov decision processes (MDP): $M = \{S, A, T, R\}$ with states $\mathbf{s} \in S$, actions $\mathbf{a} \in A$, the transition function $T : S \times A \times S \rightarrow [0, 1]$, and the reward function $R : S \times A \times S \rightarrow \mathbb{R}$. In the context of process design, the states $\mathbf{s}$ represent the flowsheet topology as well as all relevant design specifications, operating variables, thermodynamic stream data, flowrates, and compositions. The agent takes the current states $\mathbf{s}$ as input to take actions $\mathbf{a}$. These actions can include design and operating variables. In chemical processes, design variables are usually determined during the initial design phase and typically remain fixed throughout the operation, such as equipment size. Operating variables can be adjusted during operation (e.g., flow rates, and pressures). Furthermore, the actions contain discrete choices (e.g., the selection of open streams, unit operation types, or number of stages) as well as continuous choices (e.g., the length of a reactor or operating flowrates), namely hybrid action

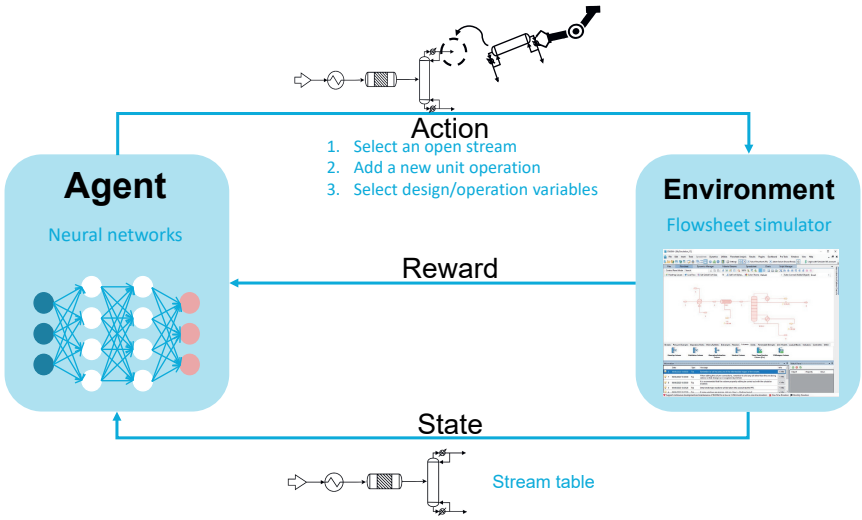


Figure 3.2: General framework of reinforcement learning for process design.

space. Usually, decisions in process design are also hierarchical. For example, the agent first determines an open stream to add a unit operation, then the type of unit operation, then design variables, and finally operating variables. After a new unit operation is added, the new flowsheet is simulated in an environment (e.g., a process simulation software). After finishing a flowsheet, a numerical reward $\mathbb{R}$ is returned to the agent. This corresponds to the objective for optimization. By repeating the design of multiple flowsheets and receiving corresponding rewards, the agent learns to design processes that maximize the reward.

In this section, we survey the RL state-of-the-art literature (summarized in Table 3.1) based on (i) information representation, (ii) agent architecture, and (iii) environment and reward.

3.2.1. INFORMATION REPRESENTATION

Chemical processes comprise various information, such as process topology, thermodynamic states, flowrates, concentrations, design variables, operating variables, components, and underlying mechanistic knowledge. The meaningful representation of the chemical process information is critical for the learning and generalization of RL agents. For RL in process design, there are currently two methods for information representations: Matrix [18–32] and graph [33, 34].

In matrix-based representation, flowsheets are represented by fixed-size matrices. Within the flowsheet matrix, the connectivity, stream compositions, thermodynamic stream data, and design variables are usually concatenated. For example,

Table 3.1: An overview of the reviewed literature and different choices of elements in RL for process design. We use Repr. to indicate information representation. Furthermore, we utilize Dis., Cont., and Hier. to denote discrete, continuous, and hierarchical action space of RL. Within the decisions of RL, we use Topo., Des. and Oper. to represent actions involved in changing flowsheet topologies, selecting design variables and operating variables, respectively.

Ref.	Repr.	Agent architecture	Environment	Action space			Decisions		
				Dis.	Cont.	Hier.	Topo.	Des.	Oper.
[22]	Matrix	Actor-Critic (SAC)	COCO	✓	✓		✓	✓	✓
[23]	Matrix	Actor-Critic (SAC)	Aspen Plus	✓	✓			✓	✓
[18]	Matrix	Actor-Critic (-)	Aspen Plus		✓				✓
[30]	Matrix	Actor-Critic (PPO)	Short-cut		✓			✓	
[24] [25] [26]	Matrix	Actor-Critic (Two players)	Short-cut	✓		✓	✓		
[28][29]	Matrix	Actor-Critic (PPO)	Short-cut	✓	✓	✓	✓	✓	✓
[20]	Matrix	Actor-Critic (ACER)	Short-cut	✓				✓	
[19]	Matrix	Policy-based (Policy search)	Short-cut		✓				✓
[21]	Matrix	Policy-based (PG)	Short-cut		✓			✓	✓
[32]	Matrix	Value-based (DQN)	IDAES	✓			✓		
[31]	Matrix	Value-based (DQN)	UniSim		✓			✓	✓
[27]	Matrix	Value-based (Q-learning)	Short-cut	✓	✓		✓	✓	
[33]	Graph	Actor-Critic (PPO)	Short-cut	✓	✓	✓	✓	✓	✓
[34]	Graph	Actor-Critic (PPO)	DWSIM	✓	✓	✓	✓	✓	✓

Göttl et al.[24–26] represented flowsheets as 16×28 matrices, where each row represents a stream and encompasses four parts: $\{\mathbf{v}, \mathbf{u}, \mathbf{d}, \mathbf{t}\}$. $\mathbf{v}_i$ has five entries, which describe the molar fractions and total molar flowrate of stream i. $\mathbf{u}_i$ stores the type of the unit operation that is downstream of stream i as one-hot encoding. Furthermore, $\mathbf{d}_i$ stores the connectivity of unit operations and has sixteen entries (i.e., this corresponds to the adjacency matrix). Finally, $\mathbf{t}_i$ has two entries: the first entry indicates whether the task is terminated (0 if not terminated), and the second entry indicates whether stream i is still unused (0 if unused). Most of the previous publications use matrix representations of flowsheet states (c.f. Table 3.1).

In graph-based representation, flowsheets are represented by directed heterogeneous graphs. Flowsheet graphs consist of nodes and edges. Unit operations are represented by nodes, also referred to as vertices $v \in V$, and streams are represented by edges $e_{vw} \in E$ connecting two nodes v and w. Importantly, node feature vector $\mathbf{f}^v \in F^V$, and edge feature vector $\mathbf{f}^{e_{vw}} \in F^E$ are associated with each node and edge, respectively. Within the node feature vectors, types of unit operation, design specifications, and operating points are encoded. The edge feature vectors contain thermodynamic states, concentrations, and flowrates. In the past, only our previous works used graph representations of flowsheets for RL [33, 34].

The comparison between flowsheet matrices and flowsheet graphs is still an open research question in the context of RL in process design. Flowsheet matrices are easier to implement than flowsheet graphs and are used by the majority of the

literature as shown in Table 3.1. Flowsheet matrices are processed by RL agents using multilayer perceptrons (MLPs) or convolutional neural networks (CNNs). However, every flowsheet graph has $N!$ different adjacency matrices. CNNs and MLPs are not permutation equivariance

$$f(\mathbf{P}^T\mathbf{x}\mathbf{P}) \neq \mathbf{P}^T f(\mathbf{x})\mathbf{P}$$

where $\mathbf{P}$ is a permutation matrix, $\mathbf{x}$ is the input matrix and f is a MLP or CNN [35]. This means that such models depend on the arbitrary order of rows/columns in the flowsheet matrix and thus, cannot generalize over flowsheet topologies. Also, the neighborhood of an entry in the matrix does not correspond to physical connectivity which makes learning using MLPs/CNNs more difficult as it requires learning long-range interactions. In contrast, (message-passing) graph convolutional networks (GCNs) are permutation equivariance and they learn from the actual connectivity of flowsheet graphs. Furthermore, MLPs and CNNs require fixed-size inputs, e.g., a pre-defined maximum number of unit operations and streams, while GCNs are size-independent [36].

3.2.2. AGENT ARCHITECTURE

RL agents consist of two components: A policy and a learning algorithm. The policy describes the behavior of the agent, mapping the current state $\mathbf{s}$ into an action $\mathbf{a}$: $\pi(\mathbf{s}) = \mathbf{a}$. It is parameterized by function approximators such as MLPs. The learning algorithm is used to continuously update the policy based on the actions, states, and rewards. Depending on the learning algorithms, the agent can be characterized into three types: Value-based, policy-based, and actor-critic-based (AC).

Value-based agents learn a functional approximator of the value function ($V_\pi(\mathbf{s})$) to take actions. The value function outputs the expected returns after the current process step t given a state $\mathbf{s}$ and a policy π: $V_\pi(\mathbf{s}) = \mathbb{E}_\pi[G_t|\mathbf{s}]$, where returns G_t:

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^k R_{t+k+1} = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{3.1}$$

where γ^k are the discount rates to determine the present value of future rewards, and k is the process step from t to the end of the episode. Similarly, we can also derive the state-action value function, namely the quality function Q_π, which calculates the expected returns given a state $\mathbf{s}$ and action $\mathbf{a}$, following policy π: $Q_\pi(\mathbf{s},\mathbf{a}) = \mathbb{E}_\pi[G_t|\mathbf{s},\mathbf{a}]$. Depending on the calculated V-/Q-value, different search algorithms such as best-first search or nearest neighbors are used to choose the final action. In the context of RL in process design, three works [27, 31, 32] deployed Q-learning based agent to perform process synthesis tasks. However, traditional

value-based agents can only take discrete actions, which hinders further development because continuous decision-making of design or operating variables is vital in process design tasks.

Policy-based agents directly learn a functional approximator of the policy function. Specifically, the policy approximator π_θ maps the current states $\mathbf{s}$ to the actions $\mathbf{a}$: $\pi_\theta(\mathbf{s}) = \mathbf{a}$. And the optimal policy π^* is obtained by maximizing the expected return $\mathbb{E}_\theta[G_t]$ through policy gradient or policy-search methods. In the context of RL in process design, Sachio et al. [21] and Perera et al. [19] utilized policy gradient methods and policy-search methods to perform process design tasks, respectively. Compared to the value-based approach, the policy-based agent can handle both discrete and continuous actions. However, policy-based methods are known for high variance and sub-optimal local solutions [37].

AC agents combine the advantages of value-based and policy-based methods. AC consists of an actor, working as a functional approximator of the policy function, and a critic, serving as a functional approximator of the value function. Therefore, AC agents explicitly optimize both value and policy functions and are able to process both discrete and continuous action spaces. In the context of RL in process design, different types of AC agents have been used such as Proximal Policy Optimization (PPO) [29] [30] [33] [34], Soft Actor-Critic (SAC) [22, 23], Two-player game [24–26], and Sample Efficient Actor Critic with Prioritized Experience Replay (ACER) [20].

The choice of agent architecture for RL in process design is an open question. AC RL is deemed to be a viable option because it combines the advantages of value-based and policy-based and can handle both discrete and continuous decisions. Specifically, PPO is the most popular algorithm in process design tasks with the advantage of less complicated implementation and a stable learning process. However, PPO is an on-policy algorithm which means the optimized policy is the same as the policy for action selection. Therefore, PPO is less data-efficient than off-policy algorithms, such as SAC and ACER, which may take less time and fewer training episodes. Moreover, AC RL comes with challenges, including complex implementation, computational demands when optimizing both actor and critic networks concurrently, and potential convergence issues [13, 37].

3.2.3. ENVIRONMENT AND REWARD

The environment simulates the processes and computes a reward as feedback to the agent. Selecting an appropriate accuracy level for the environment is a vital task for RL in process design and depends on the task-specific requirements and modeling intent. There are two main levels of accuracy: Shortcut and rigorous simulators. Shortcut simulators utilize approximated process models to ensure tractability but can be inaccurate. Rigorous simulators involve more accurate

process models that require longer computation times, as previous studies indicated [23, 34]. In the past, RL for process design has used multiple process simulation software including open-source (DWSIM, IDEAS), non-commercial (COCO), and commercial (UniSim, Aspen Plus) alternatives. Additionally, Seidenberg et al. [29] leveraged knowledge graphs to retrieve information about the design task, process knowledge, and the current state of the process. Notably, this knowledge graph was part of a manually-implemented environment and not directly accessible to the RL agent. Thus, the agent also relied on a flowsheet matrix representation as states.

Previous research optimized towards a single economic objective [18, 20–27, 29, 33, 34]. Also, some works integrate purity, recovery, power consumption, and product flow rate into scalar reward functions [30–32].

3.3. PERSPECTIVES

Despite the first demonstrations of RL for process design, it is still unclear if RL outperforms existing design methods. In our view, the big research challenge is the generalization of RL models and the use of its inference capabilities. The training phase of current RL frameworks is essentially used like a derivative-free optimization approach (e.g., a genetic algorithm) to optimize the process topology for one particular case study. Thus, a re-training is needed for a new case study and the agent fails to transfer its learning to new situations. In general, deep RL has an inference and a high learning capacity. The derivative-free optimization approach with RL does not use the full potential of RL. However, useful application of inference requires generalization across multiple case studies. This generalization requires an extension of the information representation and agent architecture to account for process-relevant knowledge. This includes domain expertise, prior process data, and physical constraints which are typically employed by engineers when designing chemical processes. Integrating this information would allow the RL agent to see what "drives the process" and ultimately unlock the full potential of RL by learning from multiple processes. We envision that RL will generalize (to some extent) and ultimately design processes at inference time. In this section, we provide our perspectives on underlying challenges and a number of other promising future works.

3.3.1. INFORMATION REPRESENTATION

Information representation is critical for RL since it encapsulates the current state of the environment, which directly affects decision-making for agents. However, current information representations still lack mechanistic knowledge and relevant process information. Furthermore, it neglects the appropriate representation of molecules. Integrating the above information will significantly benefit the

RL agent in generalization. Numerous representation methods could be potentially incorporated into RL in process design for process and molecular information representation. For example, Simplified Flowsheet Input-Line Entry-System (SFILES) [38], SFILES 2.0 [39], eSFILES [40], and knowledge graphs [29] could be used to enhance process representations. Similarly, molecular descriptors [41], molecular graphs [42], SMILES [43], knowledge graphs [44], and hypergraphs [45] could be used to encode molecular information.

3.3.2. AGENT ARCHITECTURE

In this section, we identify the limitations and potential improvements of the current agent architecture.

INTEGRATION OF MECHANISTIC KNOWLEDGE

Current RL algorithms are not sufficient to transfer knowledge into the new processes because RL agents have a limited understanding of mechanistic knowledge and physical properties. Future work could consider implementing a physics-informed RL agent by encoding information-rich representations such as knowledge graphs or hypergraphs to inform the agent. Furthermore, fundamental concepts, such as thermodynamic driving forces (Gibb's free energy), could be included in the RL agents. This allows the agent to learn general concepts that can be translated into other problems because they are based on physics.

INTEGRATION OF PRIOR DATA

RL for process design is currently initialized randomly, which can lead to suboptimal solutions, excessive training times, and frequent convergence issues. Meanwhile, there is a large number of existing digitized chemical process data from simulation files and images [46], which can potentially accelerate the learning process of the agent. Transfer learning improves learning performance by transferring knowledge from different but relevant domains [47]. In RL for process design, three work [21, 32, 34] already leveraged transfer learning to accelerate the learning process, e.g., from short-cut simulators to rigorous simulators [34] and from one case study to another case study [32]. However, in the current transfer learning setting, the agent is still not learning from existing chemical process information. Future work can consider leveraging encoder-decoder models such as Variational Autoencoders (VAEs) or transformers to learn from existing flowsheets and then applying transfer learning to the agent.

STOCHASTIC DECISION-MAKING

Considering the uncertainty of energy/feedstock prices and demand is a major challenge for renewable processes [7]. However, current RL agents ignore fluctuations in energy/feedstock prices and demand. Future research could separate

design and operating variables in the RL agent. This allows the inclusion of multiple scenarios for flexible operation. Besides, additional encoders or actors can be included to process stochastic energy prices, demands, and raw material compositions as additional inputs at an operational level. Therefore, the agent can automatically select the operating variables based on stochastic energy prices and demand in two-stage stochastic programming settings.

CONSTRAINED DECISION-MAKING

Constrained decision-making is crucial for RL in process design to ensure optimal and safe performance. However, standard RL agent frameworks cannot enforce constraints but include constraints as "soft" penalties in the reward functions [23, 33, 34]. Future work should focus on integrating constraints directly in the agent structure. As a first step, an additional critic network could be built to account for safety constraints, guiding RL agents to explore appropriate regions in policy optimization [48].

3.3.3. ENVIRONMENT AND REWARD

In this section, we offer our perspectives on the limitations of the environment and reward setup and provide several suggestions for future work.

STANDARDIZED SIMULATION INTERFACES

RL agents frequently interact with process simulators during the training process and the interaction relies on individual interfaces as Table 3.1 shows. However, current interfaces are usually simulator-specific, which means that a new interface needs to be implemented from scratch whenever a new process simulator is included. This process is highly repetitive and inefficient, especially for incorporating multi-fidelity process models. Future work could implement a standardized simulation interface that enables the agent to exchange data efficiently and uniformly between different process simulators. This interface could potentially make use of existing standards such as CAPE-OPEN [49] and DEXPI+ [50].

MULTI-FIDELITY PROCESS MODELS

Current research only leverages a single fidelity model for RL in process design tasks. However, RL agents greatly benefit from pre-training on low-fidelity process simulators and subsequent fine-tuning on high-fidelity process simulators [34]. Therefore, future research can focus on developing an agent that can dynamically select between multiple fidelity models during training. Specifically, a probabilistic model can be developed to guide the RL actor based on multi-fidelity critics to reduce training times and resolve convergence issues.

MULTI-OBJECTIVE REWARDS

Current RL frameworks for process design are not suitable for sustainable process design because they are limited to a single objective function. In the future, the current agent structures could be extended to include multiple objectives. For example, the critic network could predict multiple rewards, which will be processed by multi-objective optimization to generate corresponding weights for each objective (e.g. economic, environmental, safety) [51].

3.3.4. INTEGRATED MOLECULAR AND PROCESS DESIGN

Current RL frameworks lack the co-design of molecules. However, the design or selection of molecules is a critical task in many process design tasks, e.g., co-design of working fluids, solvents, or products [52]. Also, RL has already been used for molecular design [53]. Therefore, future work should consider integrating these concepts using RL. For instance, future work could first use an RL agent for molecular design (e.g., based on [53]) to design a solvent. Then, a mechanistic or data-driven model [42] can be used to estimate the relevant properties of the generated molecules. Subsequently, the properties are utilized to simulate the process within the RL for the process design framework. This essentially adds a new hierarchy level to the existing RL for the process design framework.

3.3.5. INTEGRATED PROCESS OPERATION AND DESIGN

Integrating process design and process control becomes increasingly relevant as renewable energy and feedstock demand fluctuates. For example, the design of a process could be optimized while simultaneously optimizing its operation under changing feedstock compositions or energy prices. Another example is the optimal design of batch processes while considering optimal operation strategies. These problems are usually formulated as mixed-integer dynamic optimization (MIDO) problems which are difficult to solve. RL is a tool to solve discrete-time dynamic optimization problems, which makes it suitable for integrating process operation and design. However, current RL works only consider process design and operation separately or focus on a specific unit operation design and operation [21]. Future research could integrate process design with process operation through the RL. For instance, future work could extend the hierarchical RL agent into separate design and operation agents. Then, the design agent defines the design variables, and the operation agent subsequently optimizes operating variables given the current design. Notably, this would require the use of a dynamic simulation environment and will lead to high computational demands. Thus, future research is needed to solve the resulting multi-scale problem efficiently.

3.3.6. BENCHMARKING WITH ESTABLISHED METHODS

Numerous established methods are available to solve design optimization problems, including deterministic (e.g., BARON or MAiNGO) and stochastic solvers (e.g., genetic algorithms, Bayesian optimization) [54]. It is still questionable how RL compares against these traditional approaches for steady-state process design. Dynamic solution approaches such as RL can in principle be used to solve static optimization problems but are likely significantly less efficient. However, many process design problems in chemical engineering are actually (mixed-integer) dynamic optimization problems (c.f. Section 3.3.5). In such instances, RL may be an efficient solution approach.

Future work should carefully assess the advantages and disadvantages of using RL for steady-state process design and static optimization in general. We recommend conducting comparisons to benchmark different methods. Moreover, we envision the development of new ML-based algorithms that integrate some of the advantages of RL (e.g., large learning capacity and inference capabilities) in the context of process design. For example, encoder-decoder models could be combined with active learning to predict process flowsheet graphs directly [8].

3.4. CONCLUSIONS

We reviewed the state-of-the-art RL in process design in terms of information representation, agent architecture, environment, and reward. RL has shown initial promising results for process design but its suitability in static process design still needs to be examined. Additionally, a detailed comparison with existing process design methods is missing and current RL frameworks show limited generalization capabilities. Therefore, we advocate that future research should benchmark RL with other process design methods. Additionally, to unlock the full potential of RL, new concepts for meaningful information representation are required. Furthermore, the integration of mechanistic knowledge, existing process data, uncertainties, and constraints would be highly beneficial for optimal decision-making. Finally, future RL frameworks could also integrate molecular design and process operation into the conceptual process design.

REFERENCES

[1] S. I. Meramo-Hurtado and A. D. González-Delgado, *Process synthesis, analysis, and optimization methodologies toward chemical process sustainability,* Industrial & Engineering Chemistry Research **60**, 4193 (2021).

[2] E. Martinez-Hernandez, *Trends in sustainable process design—from molecular to global scales,* Current Opinion in Chemical Engineering **17**, 35 (2017).

[3] T. Umeda, *Computer aided process synthesis,* Computers & Chemical Engineering **7**, 279 (1983).

[4] A. C. Dimian and C. S. Bildea, *Chemical process design: Computer-aided case studies* (John Wiley & Sons, 2008).

[5] T. Yee and I. Grossmann, *Simultaneous optimization models for heat integration—II. heat exchanger network synthesis,* Computers & Chemical Engineering **14**, 1165 (1990).

[6] L. Mencarelli, Q. Chen, A. Pagot, and I. E. Grossmann, *A review on superstructure optimization approaches in process system engineering,* Computers & Chemical Engineering **136**, 106808 (2020).

[7] A. Mitsos, N. Asprion, C. A. Floudas, M. Bortz, M. Baldea, D. Bonvin, A. Caspari, and P. Schäfer, *Challenges in process optimization for new feedstocks and energy sources,* Computers & Chemical Engineering **113**, 209 (2018).

[8] E. Hirtreiter, L. S. Balhorn, and A. M. Schweidtmann, *Toward automatic generation of control structures for process flow diagrams with large language models,* AIChE Journal (2023), 10.1002/aic.18259.

[9] T. Nabil, J.-M. Commenge, and T. Neveux, *Generative approaches for the synthesis of process structures,* in *Computer Aided Chemical Engineering*, Vol. 49 (Elsevier, 2022) pp. 289–294.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, *Playing atari with deep reinforcement learning,* (2013).

[11] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaskowski, *ViZDoom: A doom-based AI research platform for visual reinforcement learning,* in *2016 IEEE Conference on Computational Intelligence and Games (CIG)* (IEEE, 2016).

[12] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, *A general reinforcement learning algorithm that masters chess, shogi, and go through self-play,* Science **362**, 1140 (2018).

[13] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction* (MIT press, 2018).

[14] J. Hoskins and D. Himmelblau, *Process control via artificial neural networks and reinforcement learning,* Computers & Chemical Engineering **16**, 241 (1992).

[15] S. Spielberg, R. Gopaluni, and P. Loewen, *Deep reinforcement learning approaches for process control,* in *2017 6th International Symposium on Advanced Control of Industrial Processes (AdCONIP)* (IEEE, 2017).

[16] C. D. Hubbs, C. Li, N. V. Sahinidis, I. E. Grossmann, and J. M. Wassick, *A deep reinforcement learning approach for chemical production scheduling,* Computers & Chemical Engineering **141**, 106982 (2020).

[17] Y. H. Lee and S. Lee, *Deep reinforcement learning based scheduling within production plan in semiconductor fabrication,* Expert Systems with Applications **191**, 116222 (2022).

[18] J. Chen and F. Wang, *Cost reduction of CO2 capture processes using reinforcement learning based iterative design: A pilot-scale absorption–stripping system,* Separation and Purification Technology **122**, 149 (2014).

[19] A. Perera, P. Wickramasinghe, V. M. Nik, and J.-L. Scartezzini, *Introducing reinforcement learning to the energy system design process,* Applied Energy **262**, 114580 (2020).

[20] C. Caputo, M.-A. Cardin, P. Ge, F. Teng, A. Korre, and E. A. del Rio Chanona, *Design and planning of flexible mobile micro-grids using deep reinforcement learning,* Applied Energy **335**, 120707 (2023).

[21] S. Sachio, M. Mowbray, M. M. Papathanasiou, E. A. del Rio-Chanona, and P. Petsagkourakis, *Integrating process design and control using reinforcement learning,* Chemical Engineering Research and Design **183**, 160 (2022).

[22] L. I. Midgley, *Deep reinforcement learning for process synthesis,* (2020), 10.48550/ARXIV.2009.13265, arXiv:2009.13265 [cs.LG] .

[23] S. C. P. A. van Kalmthout, L. I. Midgley, and M. B. Franke, *Synthesis of separation processes with reinforcement learning,* (2022), 10.48550/ARXIV.2211.04327, arXiv:2211.04327 [cs.LG] .

[24] Q. Göttl, D. G. Grimm, and J. Burger, *Automated synthesis of steady-state continuous processes using reinforcement learning,* Frontiers of Chemical Science and Engineering **16**, 288 (2021).

[25] Q. Göttl, Y. Tönges, D. G. Grimm, and J. Burger, *Automated flowsheet synthesis using hierarchical reinforcement learning: Proof of concept,* Chemie Ingenieur Technik **93**, 2010 (2021).

[26] Q. Göttl, D. G. Grimm, and J. Burger, *Using reinforcement learning in a game-like setup for automated process synthesis without prior process knowledge,* in *Computer Aided Chemical Engineering* (Elsevier, 2022) pp. 1555–1560.

[27] A. Khan and A. Lapkin, *Searching for optimal process routes: A reinforcement learning approach,* Computers & Chemical Engineering **141**, 107027 (2020).

[28] A. A. Khan and A. A. Lapkin, *Designing the process designer: Hierarchical reinforcement learning for optimisation-based process design,* Chemical Engineering and Processing - Process Intensification **180**, 108885 (2022).

[29] J. R. Seidenberg, A. A. Khan, and A. A. Lapkin, *Boosting autonomous process design and intensification with formalized domain knowledge,* Computers & Chemical Engineering **169**, 108097 (2023).

[30] S. J. Plathottam, B. Richey, G. Curry, J. Cresko, and C. O. Iloeje, *Solvent extraction process design using deep reinforcement learning,* Journal of Advanced Manufacturing and Processing **3** (2021), 10.1002/amp2.10079.

[31] S. Kim, M.-G. Jang, and J.-K. Kim, *Process design and optimization of single mixed-refrigerant processes with the application of deep reinforcement learning,* Applied Thermal Engineering **223**, 120038 (2023).

[32] D. Wang, J. Bao, M. Zamarripa-Perez, B. Paul, Y. Chen, P. Gao, T. Ma, A. Noring, A. Iyengar, D. Schwartz, E. Eggleton, Q. He, A. Liu, O. Marina, B. Koeppel, and Z. Xu, *Reinforcement learning for automated conceptual design of advanced energy and chemical systems,* (2022), 10.21203/rs.3.rs-2248780/v1.

[33] L. Stops, R. Leenhouts, Q. Gao, and A. M. Schweidtmann, *Flowsheet generation through hierarchical reinforcement learning and graph neural networks,* AIChE Journal **69** (2022), 10.1002/aic.17938.

[34] Q. Gao, H. Yang, S. M. Shanbhag, and A. M. Schweidtmann, *Transfer learning for process design with reinforcement learning,* in *Computer Aided Chemical Engineering* (Elsevier, 2023) pp. 2005–2010.

[35] W. L. Hamilton, *Graph Representation Learning* (Springer International Publishing, 2020).

[36] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, *Graph neural networks: A review of methods and applications,* AI Open **1**, 57 (2020).

[37] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, *Bridging the gap between value and policy based reinforcement learning,* in *Advances in Neural Information Processing Systems*, Vol. 30, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, Inc., 2017).

[38] L. d'Anterroches, *Process Flowsheet Generation & Design through a Group Contribution Approach* ([CAPEC], Department of Chemical Engineering, Technical University of Denmark, 2005).

[39] G. Vogel, E. Hirtreiter, L. S. Balhorn, and A. M. Schweidtmann, *SFILES 2.0: an extended text-based flowsheet representation,* Optimization and Engineering (2023), 10.1007/s11081-023-09798-9.

[40] V. Mann, R. Gani, and V. Venkatasubramanian, *Intelligent process flowsheet synthesis and design using extended SFILES representation,* in *Computer Aided Chemical Engineering* (Elsevier, 2023) pp. 221–226.

[41] R. Todeschini and V. Consonni, *Molecular descriptors,* Recent Advances in QSAR Studies , 29 (2010).

[42] A. M. Schweidtmann, J. G. Rittig, A. König, M. Grohe, A. Mitsos, and M. Dahmen, *Graph neural networks for prediction of fuel ignition quality,* Energy & Fuels **34**, 11395 (2020).

[43] D. Weininger, *SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules,* Journal of Chemical Information and Computer Sciences **28**, 31 (1988).

[44] Y. Fang, Q. Zhang, H. Yang, X. Zhuang, S. Deng, W. Zhang, M. Qin, Z. Chen, X. Fan, and H. Chen, *Molecular contrastive learning with chemical element knowledge graph,* Proceedings of the AAAI Conference on Artificial Intelligence **36**, 3968 (2022).

[45] H. Kajino, *Molecular hypergraph grammar with its application to molecular optimization,* (2018), 10.48550/ARXIV.1809.02745, arXiv:1809.02745 [cs.LG] .

[46] L. S. Balhorn, Q. Gao, D. Goldstein, and A. M. Schweidtmann, *Flowsheet recognition using deep convolutional neural networks,* in *Computer Aided Chemical Engineering* (Elsevier, 2022) pp. 1567–1572.

[47] K. Weiss, T. M. Khoshgoftaar, and D. Wang, *A survey of transfer learning,* Journal of Big Data **3** (2016), 10.1186/s40537-016-0043-6.

[48] Q. Yang, T. D. Simão, S. H. Tindemans, and M. T. J. Spaan, *Safety-constrained reinforcement learning with a distributional safety critic,* Machine Learning **112**, 859 (2022).

[49] M. Jarke, J. Koeller, W. Marquardt, L. von Wedel, and B. Braunschweig, *Cape-open: Experiences from a standardization effort in chemical industries,* in *Proc. of 1st IEEE Conference on Standardisation and Innovation in Information Technology (SIIT 99)(Aachen, Germany)* (1999) pp. 25–35.

[50] M. Theissen and M. Wiedau, *Dexpi p&id specification,* Version 0.11 (2016).

[51] C. Liu, X. Xu, and D. Hu, *Multiobjective reinforcement learning: A comprehensive overview,* IEEE Transactions on Systems, Man, and Cybernetics: Systems **45**, 385 (2015).

[52] P. Rehner, J. Schilling, and A. Bardow, *Molecule superstructures for computer-aided molecular and process design,* Molecular Systems Design & Engineering **8**, 488 (2023).

[53] M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen, *Molecular de-novo design through deep reinforcement learning,* Journal of Cheminformatics **9** (2017), 10.1186/s13321-017-0235-x.

[54] F. Boukouvala, R. Misener, and C. A. Floudas, *Global optimization advances in mixed-integer nonlinear programming, MINLP, and constrained derivative-free optimization, CDFO,* European Journal of Operational Research **252**, 701 (2016).

III

GRAPH MACHINE LEARNING FOR MOLECULAR PROPERTY PREDICTION

4

ENVIRONMENTAL IMPACTS PREDICTION USING GRAPH NEURAL NETWORKS ON MOLECULAR GRAPHS

The chemical industry needs to undergo a significant transformation towards more sustainable and circular production systems. To guide this transformation, estimating the environmental impacts of chemical production at early product screening or development stages is highly desirable. This study leverages the molecular structure of the process products with graph neural networks (GNNs) for early-stage environmental impact approximation of chemical processes. Specifically, we use end-to-end GNN models to predict fifteen environmental impact categories, utilizing a CarbonMinds dataset of 51,905 processes producing 761 molecules produced in 91 countries, augmented with country-specific energy mix data. Our analysis begins with a comparison of Quantitative Structure-Property Relationship (QSPR) and GNN models for the climate change impact category. Specifically, we develop three different GNN models: (i) GNN with only molecular structure, (ii) GNN with molecular structure and additional geographical features, and (iii) GNN with molecular structure and additional energy mix features. The results indicate that the three GNN models show an improvement over the QSPR models. Furthermore, benchmarking our GNN models against the existing literature in the climate change impact category reveals that our models perform comparably. We then extend our approach by developing both single- and multi-task GNN models to predict all fifteen impact categories. The findings indicate that multi-task learning can improve model performance in complex environmental impact predictions compared to single-task GNNs. Therefore, we recommend using a multi-task GNN for predicting multiple impact categories, with single-task models applied to fine-tune performance on underperforming categories. Although our proposed approach shows improvements over previous models, the prediction of environmental impacts solely based on molecular information remains a rough approximation.

This chapter has been published as "Environmental impacts prediction using graph neural networks on molecular graphs" by Gao, Q., Balhorn, L.S., Laera, A., Meys, R., Goßen, J., Weber, J.M., Wernet, G. & Schweidtmann, A.M., Computers & Chemical Engineering, 109362, (2025)

4.1. INTRODUCTION

The chemical industry is currently in need of a significant paradigm shift towards more sustainable and circular processes. This transition requires reducing the environmental impact associated with existing chemical production processes. In this context, it is critical to recognize that modifications in the design of chemical products and processes at early technology readiness levels (TRLs) [1] are most feasible and cost-effective. The role of computational prescreening techniques [2–7] is increasingly important, as these methods offer a systematic approach to assessing the environmental impact of chemical production processes and their viable alternatives during the early stage of development.

The state-of-the-art methodology for determining the environmental impact of processes and products is Life Cycle Assessment (LCA). LCA is an ISO-normed methodology for assessing the environmental impact of a product or process [8]. The core principle of LCA is to trace the principal stages and processes in a product's lifecycle, from raw material extraction through manufacturing, usage, and recycling, to eventual disposal, pinpointing and calculating the environmental impacts at each stage. Because of its holistic nature, LCA requires detailed information about the processes and products across all phases of the life cycle inventory—often from cradle-to-gate and beyond. Specifically, in this study, we focus exclusively on the cradle-to-gate phase. The significant data demands, spanning multiple levels of the supply chain of products, often present a major obstacle to conducting a thorough LCA. This challenge is particularly evident in the chemical industry and its supply chains, which are characterized by complex process networks with interdependent exchanges of energy, mass, and water, as well as the use of diverse intermediate products and feedstocks. Additionally, the high data demand for a full LCA study often hinders early-stage screening studies on multiple product and/or process alternatives. Therefore, to address these data gaps, LCA practitioners have deployed simplified approaches [9–12], for example, streamlined LCA [9], to overcome the data scarcity problem.

Streamlined LCA uses simplified models, databases, and assumptions to estimate impacts without extensive data collection and analysis as in full LCAs. The streamlined LCA methods consist of two parts: (1) the input data and (2) the analysis model. The data is usually collected from key aspects such as the molecular structure and it can be characterized by the technology readiness level as defined in Buchner et al [1]. Several databases [13] provide access to LCA impact category data, including Ecoinvent [14] and the European Reference Life Cycle Database (ELCD) [15]. These resources offer valuable data for estimating environmental impacts across various categories. The analysis model maps the input data to output environment impact categories through a predictive model. Molecular structure models (MSMs) have been widely used in assessing chemical products [16, 17].

The rationale of MSMs is that the molecular structure encodes key information that has a direct impact on (i) the intricacy of their production process and (ii) their potential for danger and degradation at the end of life, which in turn affects aspects of their life cycle environmental impact. Unlike the prediction of conventional physicochemical properties [18, 19], such as boiling point or solubility, predictions of LCIA impact scores aim to assess properties of an entire industrial supply chain, including aspects of production processes and specific process conditions. This complexity introduces greater variability and noise into the data, making accurate predictions more challenging. Various modeling approaches have been explored for predicting LCA impact categories within MSMs. These approaches can be broadly classified based on molecular representation methods and the techniques used to map these representations to corresponding impact category values. Molecular representations generally fall into three categories: molecular descriptors, molecular graphs, and molecular strings. Techniques for mapping these representations to impact values range from simple linear regressions to advanced machine learning models, including large language models, with training parameters varying from a few thousand to billions.

Quantitative structure-property relationship (QSPR) models [20], utilizing molecular descriptors, as a type of MSMs have been successfully deployed to predict LCA impact categories in the previous literature. QSPRs characterize molecules with various structural, chemical, physical, and biological features, group contributions [18, 21], referred to as molecular descriptors that are then mapped to a property of interest by a linear or nonlinear model, such as support vector machines, decision trees, random forests, and multilayer perceptrons (MLPs). For example, Wernet et al. [22, 23] first utilized both linear regression models and MLPs for molecular structures to predict several impact categories such as cumulative energy demand (CED), global warming potential (GWP), the biological and chemical oxygen demands (BOD and COD), the total organic carbon (TOC), the Ecoindicator'99(H/H): human health (HH), ecosystem quality (EQ), resources (R) and total (T) scores, achieving R^2 values from 0.41 to 0.69 across respective categories. Later, Song et al. [24] also utilized molecular descriptors and MLPs to predict six impact categories, including CED, GWP, HH, EQ, T, and acidification (AC). The R^2 values ranged from 0.45 to 0.87. Sun et al. [25] further utilized four common machine learning algorithms: support vector machines, decision trees, random forests, and MLPs to predict six impact categories, and the R^2 values ranged from 0.73 to 0.86. Additionaly, Baxevanidis et al [26] deployed six different linear and nonlinear regression models to predict CED, GWP, and Ecoindicator'99. The R^2 values ranged from 0.08 to 0.26.

While predicting environmental impacts directly from molecular descriptors is an attractive approach, it presents challenges in achieving consistently accurate

and generalizable models. Specifically, when using only product features such as molecular structure, these models most likely struggle to accurately predict aspects of the impact categories that depend on process-level decisions. This limitation arises because the models do not account for the production methods (e.g., derived from crude oil versus electrolyzers powered by green energy), which significantly influences environmental outcomes. To mitigate this problem, CalvoSerrano et al. [27–29] added thermodynamic features with molecular descriptors as input, formulated into Mixed Integer Nonlinear Programming (MINLP) problems and achieve relative errors in the range 20–44% for CED, GWP, COD, BOD, TOC, T, HH, EQ, and depletion of resources (Res). Also, Kleinekorte et al. [30] additionally included process descriptors in MLPs, together with molecular descriptors (information commonly available at TRL 2 of process/product development), to predict seventeen impact categories with R^2 between 0 and 0.66. Karka et al. [31] incorporated MLPs and decision trees to process molecular descriptors, process, and energy-related parameters, and predict twenty-three LCA metrics with R^2 between 0.50 and 0.88.

Recently, end-to-end learning, operating mainly on molecular graphs or molecular strings, has shown promising results in property prediction and molecule generation. In contrast to traditional QSPR models, the aim of end-to-end learning is to train a model to automatically learn from input data, such as molecular graphs or molecular strings, to final output predictions without the need for manual feature engineering or intermediate steps. Molecular strings with large language models [32, 33] for molecular property prediction are also a promising research direction. However, in this work, we mainly focus on the molecular graph perspective. In particular, graph neural networks (GNNs) have shown promising results in these tasks [34–38]. GNNs can learn to predict properties from molecular graphs where nodes present as atoms and edges correspond to the bonds, which constitutes an end-to-end supervised learning setup without manual feature selection [39]. GNNs reached state-of-the-art accuracy in predicting molecular properties, for regression tasks, like electron affinity [37], excitation energies [38], and classification tasks, such as toxicity of molecules [40]. In the context of environmental impact category prediction, Kleinekorte et al. [41] recently utilized a GNNs-based encoder-decoder network for processing molecular graphs. In addition, they derived process descriptors from the stoichiometric reaction equation and combine molecular and process descriptors in a Gaussian process to predict the global warming impact (GWI). The model obtained an R^2 value of 0.53, demonstrating the potential of GNNs for learning molecular descriptors for environmental impact categories prediction tasks. Additionally, Zhang et al. [42] proposed FineChem 2, which is based on a GNN and a transformer framework, to assess the product carbon footprints (PCF), the same as GWI and

GWP, of chemicals.

With those advancements in utilizing GNNs in environmental impact prediction, there are still some remaining research gaps. Primarily, while GNNs have shown good results in predicting GWI (GWP or PCF), the results of other crucial impact categories like AC and HH still need to be explored. Furthermore, current GNN prediction models are made for individual impact categories separately, namely, single-task GNN. However, multi-task GNNs have outperformed single-task GNNs on several molecular prediction tasks [43–46]. Multitask learning is a machine learning approach where a model is trained simultaneously on multiple related tasks. A shared representation is first learned across multiple tasks, capturing generalizable information. This shared representation serves as a foundation, upon which individual hidden layers are utilized to extract task-specific features. Consequently, a multi-task GNN has the potential to enhance predictive accuracy in the domain of environmental impact assessment by effectively integrating and learning from multiple related tasks. Furthermore, the current literature lacks a comparative analysis between GNNs and QSPR models. To guide future research effectively, particularly in the context of model selection, a comparison between these methodologies is necessary.

We propose to utilize GNNs to estimate the environmental impact categories. First, our contribution is applying GNN-based models to predict fifteen distinct midpoint environmental impact categories—a broader scope than most previous studies, which typically focus on less than three categories. Second, we utilize a comprehensive dataset in a total of 51,905 processes comprising 761 molecules, enriched with geographic and energy mix information from 91 different countries. Additionally, those impact scores are calculated and collected by the Environmental Footprint (EF) framework developed by Carbon Minds independently outside of this work, which is based on the Environmental Footprint v3.0 [47] and is reported using a cradle-to-gate system boundary. Third, three different GNN models are proposed: 1. Molecular features only (GNN-M); 2. Molecular features and one-hot encoded country features (GNN-C); 3. Molecular features and energy mix features (GNN-E). Additionally, we establish a comparative framework by developing QSPR models using molecular descriptors from Song et al. [24] as a benchmark to evaluate the performance of GNN models, with a specific focus on the climate change (CC) category because it is the most extensively studied and standardized impact category. CC values used in our study correspond to GWI — the total life-cycle climate impact per kg of product, expressed in kg CO_2-eq. Fourth, we develop single-task GNN-C and GNN-E to predict all fifteen impact categories separately. Additionally, to further improve prediction accuracy and data efficiency, we propose multi-task GNN-C and GNN-E. These models share message-passing layers across tasks while using separate MLP heads for each

impact category, allowing shared learning across categories. Finally, we publish the trained models and code to support reusability and future research: `https://github.com/process-intelligence-research/LCA_GNNs.git`. Together, these contributions could provide data-driven tools for early-stage sustainability assessment.

4.2. PRELIMINARIES

In this section, we briefly introduce the fundamentals of QSPR and GNN modeling.

4.2.1. QUANTITATIVE STRUCTURE-PROPERTY RELATIONSHIP

QSPR is a method used in chem- and bioinformatics to establish relationships between the molecular structure of compounds and their physical or chemical properties, consisting of two parts: (i) molecule description and (ii) property regression. In step (i), the structure of a molecule m is represented by selected molecular descriptors d_i. Mathematically, this can be described as $D : m \mapsto \mathbf{d}$ with $\mathbf{d} = [d_1, d_2, ..., d_n]^T$. In the step (ii), the function $F(\cdot)$ predicts the target property $\hat{p}$ based on the descriptors d_i [48] as $\hat{p} = F(\mathbf{d}) = F(D(m))$. The function $F(\cdot)$ can be linear or nonlinear (multivariate) regression methods, such as random forests, support vector machines, or MLPs. The choice of molecular descriptors depends on the molecules and properties of interest. Two frequently used descriptor types are structural groups and molecular fingerprints. Group contribution methods (GCMs) [49, 50] decompose the molecular structure into predefined structural groups, e.g., >CH- (non-ring), or =CH- (ring), with the frequency of these groups serving as molecular descriptors. Alternatively, molecular fingerprints denote the molecules as a vector in which the molecular structure such as the count of substructures, similar to the GCM, or geometric distances between two atoms or structural groups [51] are stored. A critical challenge in QSPR modeling is selecting appropriate descriptors for specific property prediction tasks, [52] as effective property prediction relies on the use of the most informative descriptors. However, informative descriptors are not always apparent prior to model development. Therefore, to mitigate this challenge, principal component analysis (PCA) has been utilized to extract informative features from molecular descriptors. PCA is a dimensionality reduction technique that transforms correlated variables into a smaller set of uncorrelated components while preserving as much variance as possible. Song et al.[24] applied PCA to extract 60 features from a total of 3839 molecular descriptors, which were subsequently used to predict six impact categories. However, even with PCA, the initial selection of molecular descriptors remains a necessary first step, requiring manual effort. As a result, the development of predictive QSPR models often still relies on expert intuition[48, 52].

4.2.2. GRAPH NEURAL NETWORKS

GNNs are a deep learning technique designed to learn properties directly from graph representations. Molecules are represented as graphs where atoms are denoted as nodes m, also referred to as vertices $v \in V$, and bonds are represented as edges $e_{vw} \in E$ connecting two nodes v and w. Importantly, a feature vector is typically assigned to each node and each edge, which includes specific information for the atom or bond, e.g., the atom type or the bond type. Mathematically, we denote the feature vector of a node v with $\mathbf{f}^v \in F^V$ and the feature vector of an edge e_{vw} with $\mathbf{f}^{e_{vw}} \in F^E$.

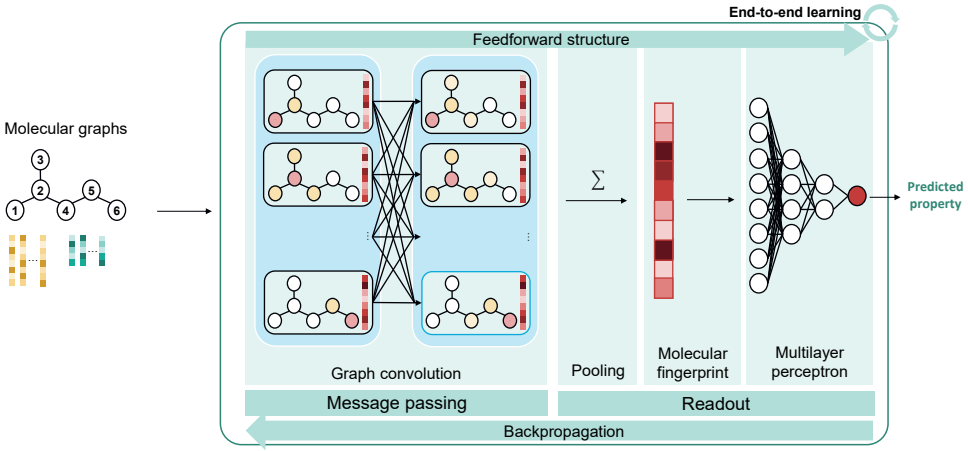


Figure 4.1: Illustration of a GNN architecture for molecular property prediction (based on [53]).

The framework of GNNs consists of two phases: The message passing phase and the readout phase as shown in Figure 4.1. The message-passing phase is responsible for extracting structural information from the molecular graph through graph convolutions. Within the graph convolution process, each node exchanges information as messages that are passed along the edges to their neighboring nodes. These messages are based on the feature vectors of the corresponding neighboring nodes and edges. For example, at the beginning of graph convolution layer l, the initiated state of a node with the corresponding feature vector is denoted as $\mathbf{h}_v^0 = f^v$ with $l = 0$. Then, the molecular graph traverses the individual graph convolutional layers $l \in \{1, 2, ..., L\}$. Within a graph convolutional layer l, the hidden state of each node is updated based on the hidden states of the corresponding neighborhood nodes and the features of the associated edges. Thereby, with a total of L graph convolutional layers, each node receives direct information

about an environment with a radius of L nodes. The updating process can be written as

$$\mathbf{m}_v^l = A_l(\{M_l(\mathbf{h}_w^{l-1}, \mathbf{f}^{e_{vw}})) \mid w \in N(v)\}) \tag{4.1}$$

$$\mathbf{h}_v^l = U_l(\mathbf{h}_v^{l-1}, \mathbf{m}_v^l) \tag{4.2}$$

The function M_l maps the *message* from one of its neighbor nodes w to node v, with $w \in N(v)$. The message is a function with two parts: (i) the hidden states of neighbor nodes w: $\mathbf{h}_w^{l-1}$, and (ii) the edge e_{vw} with the associated edge feature vector $\mathbf{f}^{e_{vw}}$. Furthermore, the aggregation function A_l aggregates the messages from all the neighbors of node v in $\mathbf{m}_v^l$. Then, the neighbor messages $\mathbf{m}_v^l$ are finally combined with the preceding hidden state of the node v itself, $\mathbf{h}_v^{l-1}$, in the update function U_l to update the hidden state of node v. Consequently, the hidden state of a node in layer l depends on its previous hidden state, the previous hidden states of its neighboring nodes, and the associated edges. With iterations of multiple graph convolutional layers L together, the information of all neighboring nodes within a distance of L nodes is passed to a node, thereby enabling learning of the so-called L-hop local environment of each node.

In the *readout phase*, a graph representation vector, the so-called molecular fingerprint vector $\mathbf{h}_G$ is aggregated from the learned structure information during message passing by the pooling function P such as $\mathbf{h}_G = P(\{\mathbf{h}_v^L \mid v \in V\})$. In particular, the pooling function P is commonly chosen as the mean, sum, or max function [54]. Finally, the molecular fingerprint $\mathbf{h}_G$ is mapped to the property of interest by a regression model such as an MLP: $\hat{p} = \text{MLP}(\mathbf{h}_G)$.

The functions in the message passing phase and pooling functions in the readout phase are both explicit and differentiable, which enables the training of the GNN with backpropagation in a supervised learning setup [55, 56]. This means that GNNs learn in an end-to-end manner, from the molecular graph to the property of interest instead of steps for the selection of informative molecular descriptors as is required in QSPR/QSAR modeling.

4.3. DATASETS

The dataset comprises 51,905 processes producing 761 distinct molecules (represented by SMILES string [57]) from 91 countries, where the target chemical is manufactured, which are incorporated as categorical features. Furthermore, each process contains 15 impact categories, as shown in Table 4.1. These impact results are process-specific, meaning the values for the same chemical product may vary depending on the production method and region. Acidification (AC) quantifies the potential of acidifying substances to disrupt ecosystems, expressed in mol H^+ equivalents. Climate change (CC) measures greenhouse gas emissions contributing to global warming in kg CO_2 equivalents. Ecotoxicity, freshwater

(ECO) assesses the toxic effects of chemicals on aquatic life, reported in comparative toxic units for ecosystems (CTUe). Energy resources, fossils (ER) reflects the depletion of non-renewable fossil fuels in megajoules. Eutrophication is divided into freshwater (EUF), marine (EUM), and terrestrial (EUT) categories, capturing nutrient enrichment impacts using phosphate (kg PO_4 eq), nitrogen (kg N eq), or mol N equivalents, respectively. Human toxicity (HT) estimates long-term effects of chemical exposure on human health in CTUh. Ionising radiation (IR) represents human exposure to radioactive substances relative to uranium-235 (c kBq U-235 eq). Land use (LU) captures impacts on soil quality and ecosystem functionality. Material resources, metals/minerals (MR) quantifies the depletion of abiotic mineral and metal resources in kg Sb equivalents. Ozone depletion (OD) reflects the contribution to stratospheric ozone layer degradation in kg CFC-11 equivalents. Particulate matter formation (PMF) measures the increase in respiratory disease incidence from airborne particles. Photochemical ozone formation (POF) estimates the formation of ground-level ozone harmful to human health, reported in kg NMVOC equivalents. Lastly, water use (WU) accounts for water scarcity potential in m^3 water equivalents. These fully characterized scores require no further transformation and serve as direct prediction targets in our models. For additional details, we refer to the literature [47, 58]

All the impact categories are calculated using the Environmental Footprint (EF) framework implemented into the Carbon Minds Database (V1.01, 2022) [58]. This framework is based on the Environmental Footprint v3.0 (EF v3.0) Life Cycle Impact Assessment (LCIA) method. EF v3.0 [47] is a life cycle impact assessment method developed by the European Commission to support consistent, science-based environmental evaluation of products. It provides midpoint characterization factors across multiple impact categories and is aligned with the Product Environmental Footprint (PEF) framework for policy and industry applications. Unlike ReCiPe2016 [59], EF v3.0 follows a single, consensus-based modeling approach without offering multiple perspectives. EF v3.0 is typically applied to chemical products using a cradle-to-gate system boundary, meaning it accounts for impacts from raw material extraction up to the point of product leaving the production site. It is important to mention that EF v3.0 includes 16 categories, due to the distinction between Human Toxicity of carcinogens and noncarcinogens. In this work, we count Human Toxicity as a single category. Furthermore, the collection and precalculation of the impact scores are done by Carbon Minds independently outside of this work.

In addition, Figure 4.2 demonstrates the count of molecules in our dataset containing each functional group. Note that these are not counts of unique molecules, as a single molecule may contain multiple functional groups. Approximately 728 molecules are organic, and 33 molecules are non-organic. Specifically, the most

frequent groups were alcohols (13%), aromatics (11%), and amines (10.5%), followed by halogenated compounds, ketones, and esters. This distribution reflects a broad spectrum of functionalized molecules relevant to industrial chemical processes. Approximately 9% of molecules could not be clearly assigned to any predefined group and were labeled as “Other”.

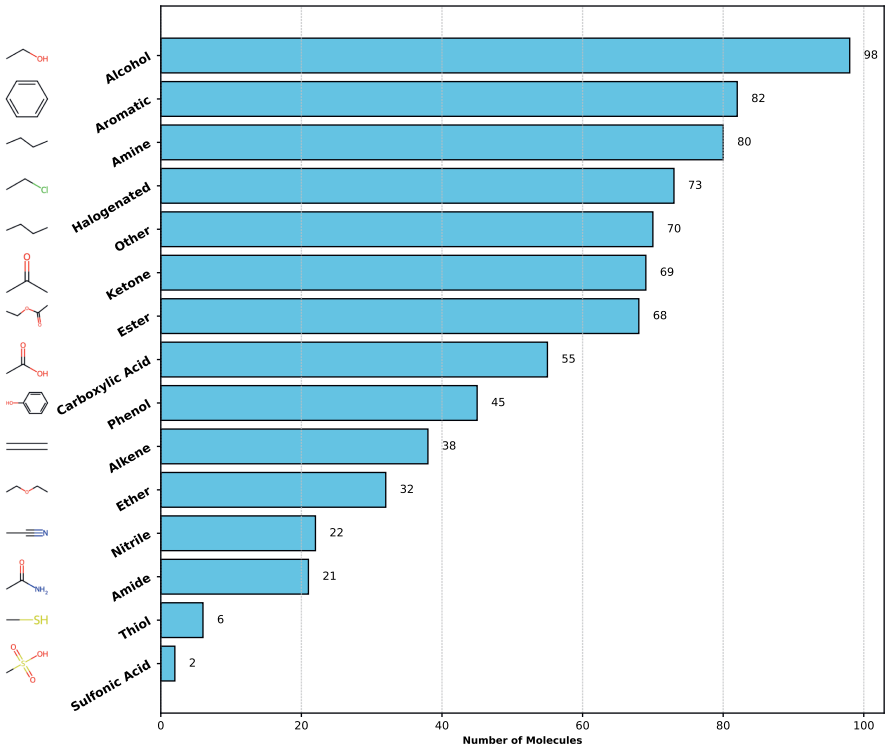


Figure 4.2: Functional group counts of the molecules in the dataset. Notably, one molecule could potentially contain more than one functional groups.

Molecular features with categorical country features offer a broad overview, however, they fall short of capturing the important contributions of different energy sources and the dynamic shifts in policy and technology that significantly influence the environmental impact categories. To address this limitation and enrich the predictive capability of our model, we integrate energy mix data that details the specific contributions of various energy sources to carbon emissions. This data is collected from the International Energy Agency (IEA) open-source version for 49 countries. The rest of the countries are assigned to broader regions (e.g., Africa, Middle East). Moreover, these data encompasses industry consumption

energy data across eight product flows: Coal, peat, and oil shale; Crude, NGL, and feedstocks; Oil products; Natural gas; Nuclear; Renewables and waste; Electricity and heat. The detailed information is shown in Table 4.8.

Table 4.1: Overview of fifteen selected impact categories. The corresponding impact scores and units are listed. Note that the term "potential" in Table 1 refers to the standard terminology of midpoint impact categories from the EF v3.0 LCIA method, which requires no further transformation.

Impact categories	Impact scores (unit)
Acidification (AC)	Accumulated exceedance (mol $H+_{eq}$)
Climate change (CC)	Global Warming Impact (kg $CO_{2\ eq}$)
Ecotoxicity, freshwater (ECO)	Comparative Toxic Unit for ecosystems (CTUe)
Energy resources, fossils (ER)	Abiotic depletion potential (MJ)
Eutrophication, freshwater (EUF)	Fraction of nutrients reaching freshwater end compartment (kg $PO_{4\ eq}$)
Eutrophication, marine (EUM)	Fraction of nutrients reaching marine end compartment (kg N_{eq})
Eutrophication, terrestrial (EUT)	Accumulated exceedance (mol N_{eq})
Human toxicity (HT)	Comparative Toxic Unit for humans (CTUh)
Ionising radiation (IR)	Human exposure efficiency relative to U^{235}_{eq} (c kBq U^{235}_{eq})
Land use (LU)	Soil quality index (-)
Material resources, metals/minerals (MR)	Abiotic depletion potential (kg Sb_{eq})
Ozone depletion (OD)	Ozone depletion potential (kg CFC-11_{eq})
Particulate matter formation (PMF)	Impact on human health (Disease incidence)
Photochemical ozone formation, human health (POF)	Tropospheric ozone concentration increase (kg $NMVOC_{eq}$)
Water use (WU)	User deprivation potential(m^3 $water_{eq}$)

4.4. METHODS

In this section, we introduce the development of the GNN model. First, we describe the data representation approaches for molecules, countries, and energy mix features in Section 4.4.1. Next, we present the GNN and QSPR architectures in Section 4.4.2. Finally, the model extensions for single/multi-task learning are described.

4.4.1. DATA REPRESENTATION

Data representation is critical since it directly affects the training process for ML models. In QSPR, molecules are denoted as molecular descriptors. Here, each molecule is described by 56 molecular descriptors as proposed in the earlier work by Song et al. [24]. These descriptors are automatically generated by the software *Dragon 7* [60]. The list of descriptors can be found in Song et al. [24] and contains common descriptors such as molecular weight, number of aromatic rings, number of functional groups, and number of halogen atoms. We represent each product molecule with a vector d of the length 56, where each entry corresponds to a descriptor with continuous or binary values. Notably, those descriptors are only utilized for QSPR models training.

In GNNs, molecules can be naturally represented by molecular graphs where atoms are denoted as nodes and edges are represented as edges. Each node and

edge has one associated feature vector, which is outlined in Table 4.2 and 4.3. These features are local graph-based representations and are fundamentally different from the global, descriptor-based features used in the QSPR model. For example, the QSPR descriptors include molecular weight, percentage of N atoms and fragment counts—each computed from the entire molecular graph—whereas the GNN models operate on atomic and bonding features such as atom type, hybridization state, and bond type, which are used in the message passing process. All the features are represented as a one-hot encoder in the corresponding dimension for this feature, where a single entry with value one at the index corresponds to the value of the feature. Specifically, molecules are given as SMILES and we convert SMILES strings into molecular graphs through RDkit [61].

Table 4.2: Atom features for initial node feature vector [53, 55, 62] in GNN models. Features are typically encoded as one-hot vectors.

Feature	Description	Dimension
Atom type	Atom type (e.g., C, H, O) ordered by atomic number	19
Is in ring	Whether the atom is part of a ring	1
Is aromatic	Whether the atom is part of an aromatic ring	1
Number of neighbours	Number of bonded atoms	6
Hybridization	Atom hybridization (e.g., s, sp, sp2...)	5

Table 4.3: Bond features for initial edge feature vector [53, 55, 62] in GNN models. Features are typically encoded as one-hot vectors.

Feature	Description	Dimension
Bond type	Bond type as in single, double, triple or aromatic	4
Is conjugated	Whether the bond is conjugated	1
Is in ring	Whether the bond is in a ring	1

4.4.2. MODEL ARCHITECTURES

We first set up a QSPR model to do the comparative study with GNN-based models. The QSPR model is adapted from [24], which contains an $\mathbf{MLP}_{\text{QSPR}}$ with two fully connected layers with 16 hidden dimensions and ReLu activation function. Taking molecular descriptors $\mathbf{d}$ as input, the QSPR model calculates the target property $\hat{\mathbf{p}}$ by Equation 4.3. The QSPR model parameters are listed in Table 4.10.

$$\hat{\mathbf{p}} = \mathbf{MLP}_{\text{QSPR}}(\mathbf{d}) \tag{4.3}$$

Figure 4.3 shows three GNN models that we propose: 1. Molecular features

only (GNN-M); 2. Molecular features and one-hot encoded country features (GNN-C); 3. Molecular features and energy mix features (GNN-E). All three GNN-based models deploy the same model architecture for molecular feature representation learning. We first convert the molecules into attributed molecular graphs that serve as input to the GNN. In the message-passing phase, gated recurrent units (GRUs) [63] are used to explore local atomic environments within the molecular graphs, which has shown promising results in molecular property prediction [53, 64, 65]. Specifically, the hidden state in layer l is updated by Equation 4.4:

$$\mathbf{h}_v^l = \mathbf{GRU}(\mathbf{h}_v^{l-1}, \sigma(\theta_v \cdot \mathbf{h}_v^{l-1} + \mathbf{m}_v^l)) \tag{4.4}$$

where $\mathbf{m}_v^l$ is given by Equation 4.5.

$$\mathbf{m}_v^l = \sum_{w \in N(v)} \mathbf{MLP}_{\theta_e}(\mathbf{f}^{e_{vw}}) \cdot \mathbf{h}_w^{l-1} \tag{4.5}$$

The edge features $\mathbf{f}^{e_{vw}}$ are first mapped by a **MLP** into a parameter matrix θ_e, which is further multiplied with the hidden states of neighborhoods of node v, $\mathbf{h}_w^{l-1}$ where $w \in N(v)$, to get message $\mathbf{m}_v^l$. The message $\mathbf{m}_v^l$ is then added to hidden state $\mathbf{h}_v^{l-1}$ multiplied with a parameter matrix θ_v. The result is passed into an activation function, exponential linear unit (ELU), and then together with hidden state $\mathbf{h}_v^{l-1}$ to get the final updated hidden state $\mathbf{h}_v^l$ after **GRU**. Furthermore, after l layers of graph convolution, the molecular fingerprint $\mathbf{h}_G$ is calculated by Equation 4.6.

$$\mathbf{h}_G = \sum_{v \in V} \mathbf{h}_v \tag{4.6}$$

where the pooling function is the sum. To note that, the initial hidden states $\mathbf{h}_v^0$ are passed into a shallow **MLP** with **ReLu** activation to ensure the uniform dimension.

All three GNN-based models use the same message passing layers to process molecular graphs but differ in how they incorporate additional features for property prediction, which are all implemented by the Pytorch-Geometric package. The detailed model parameters can be found in Table 4.11, Table 4.13, and Table 4.12 for GNN-M, GNN-C, and GNN-E in the Appendix section, respectively. It is important to mention that the concatenation of learnable molecular embedding is similar to traditional approaches that use molecular descriptors alongside thermodynamic or process descriptors to predict impact scores, as demonstrated in several previous studies [28, 30, 31]. The key distinction in our work is that the molecular representation is now learned dynamically via GNNs rather than being manually engineered.

- **GNN-M**: The molecular fingerprint $\mathbf{h}_G$ is directly fed into an MLP to predict the target property:

$$\hat{\mathbf{p}} = \mathbf{MLP}_{\text{GNN-M}}(\mathbf{h}_G) \tag{4.7}$$

- **GNN-C**: A one-hot encoded country feature $\mathbf{F}_c$ (91-dimensional) is concatenated with the molecular fingerprint before training:

$$\hat{\mathbf{p}} = \mathbf{MLP}_{\text{GNN-C}}\left[\mathbf{Cat}(\mathbf{h}_G, \mathbf{F}_c)\right] \tag{4.8}$$

- **GNN-E**: The model extends GNN-C by also incorporating a seven-dimensional energy mix feature $\mathbf{F}_e$:

$$\hat{\mathbf{p}} = \mathbf{MLP}_{\text{GNN-E}}\left[\mathbf{Cat}(\mathbf{h}_G, \mathbf{F}_e)\right] \tag{4.9}$$

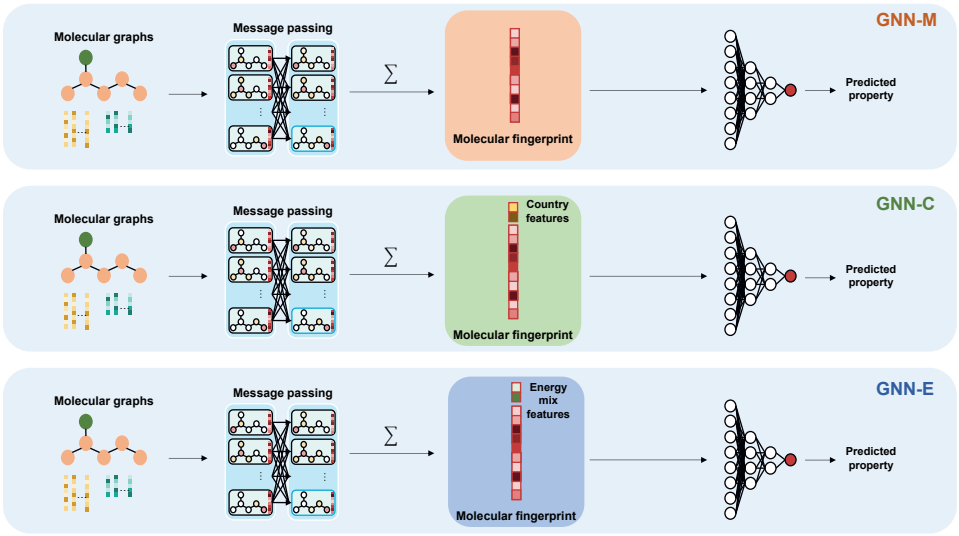


Figure 4.3: Illustration of three GNN architectures: GNN-M, GNN-C and GNN-E.

4.4.3. TRAINING APPROACH

We began by training a QSPR model and three GNN-based models (GNN-M, GNN-C, and GNN-E) for the CC impact category. The QSPR model takes molecular descriptors (listed in Table 4.9) as input and was evaluated using 10-fold cross-validation. Training hyperparameters for this model are provided in Table 4.10.

The GNN-M model was trained under the same cross-validation process, using molecular graphs as input. Details of its architecture and training settings are summarized in Table 4.11. For the GNN-C and GNN-E models, the molecular graph is learned through message passing to produce a molecular fingerprint. In GNN-C, this fingerprint is concatenated with a one-hot-encoded geographical feature vector (dimension = 91), while in GNN-E, it is concatenated with an

energy mix feature vector (dimension = 7). Both are passed through multilayer perceptrons (MLPs) for CC prediction. Architectural and training configurations for these models are shown in Table 4.13 and Table 4.12.

We then extended GNN-C and GNN-E to cover all 15 impact categories. These models were trained in both single-task and multi-task settings. In single-task training, a separate model is trained independently for each target property—this is the approach used for the initial CC models—resulting in different fingerprints for the same molecule across tasks. In contrast, multi-task learning employs a shared model where all tasks utilize the same message passing layers for molecular representation, while separate MLPs are used for each prediction target. This allows the model to learn generalizable features across tasks while still capturing task-specific signals in the readout layers.

Multi-task learning has been shown to improve generalization and reduce overfitting by encouraging shared feature learning across tasks [66]. Prior work has demonstrated its advantages in molecular property prediction [44, 67, 68]. In our case, we apply a multi-task strategy to jointly predict 15 LCA impact categories using a shared GNN backbone and 15 parallel MLP heads.

All models were trained on a dataset of 761 molecules, with 51,905 total processes across countries and impact categories, using 10-fold cross-validation. The detailed input-output structure is listed in the Table 4.4. Hyperparameters and architectural configurations for single-task GNN-C and GNN-E models are listed in Table 4.13 and Table 4.12, respectively. Learning rates were tuned individually for each category and are reported in Table 4.16. Multi-task model configurations for both GNN-C and GNN-E are detailed in Table 4.14 and Table 4.15.

To avoid overfitting and improve generalization, we apply several standard regularization techniques during model training. Specifically, we employ early stopping, which stops training when performance on the validation set ceases to improve for a fixed number of epochs, preventing overfitting to the training data. Additionally, we apply a learning rate decay strategy, which gradually reduces the learning rate over time to allow finer convergence in later training stages. All corresponding training hyperparameters are summarized in the Appendix tables. Importantly, folds are constructed based on the number of unique molecules, rather than the total number of samples. For the QSPR and GNN-M models, where only molecular descriptors or molecular graphs are used as input, each molecule corresponds to a single data point (e.g., one climate change impact value). In these cases, each fold consists of 80% of the molecules for training, 10% for validation, and 10% for testing. This ensures that each molecule is seen only once across the folds. In contrast, the GNN-C and GNN-E models incorporate additional contextual information—namely, geographical and energy mix features. Here, each molecule may be associated with multiple samples due to its presence

across different countries. However, not all chemicals have complete data for all 91 countries. As a result, when folds are split based on unique molecules, the number of training, validation, and test samples may vary across folds. The reason for splitting by unique molecules rather than by sample is to evaluate the model's ability to generalize to entirely unseen chemicals. This approach reflects a more realistic deployment scenario where the model is applied to new molecules not encountered during training.

Table 4.4: Overview of model inputs, outputs, and implementation frameworks

Model	Input	Output
QSPR model	Molecular descriptors (Dragon 7)	Climate change (CC) score
GNN-M	Molecular graphs (RDKit)	Climate change (CC) score
Single-task GNN-C / GNN-E	Molecular graphs + country or energy-mix vectors	One impact category score
Multi-task GNN-C / GNN-E	Molecular graphs + country or energy-mix vectors	15 impact category scores

4.4.4. EVALUATION METRICS

To assess model performance, we use two standard regression metrics: the mean relative error (MRE) and the coefficient of determination (R^2).

The MRE is defined as:

$$\text{MRE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|, \tag{4.10}$$

where y_i and $\hat{y}_i$ denote the ground truth and predicted values, respectively, and n is the number of samples. MRE provides a relative measure of prediction error, normalized by the true values, and is robust to scale differences across categories.

The R^2 score is given by:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \tag{4.11}$$

where $\bar{y}$ is the mean of the ground truth values. R^2 quantifies the proportion of variance in the target variable that is explained by the model. An R^2 value of 1 indicates perfect prediction, while a value of 0 means the model performs no better than predicting the mean.

Notably, R^2 can be negative when the model performs worse than the baseline prediction $\hat{y}_i = \bar{y}$. A negative R^2 suggests that the model fails to capture the underlying trend in the data and instead increases the prediction error relative to a naive mean-based approach. Note that throughout this chapter, we use the coefficient of determination (R^2) rather than correlation coefficients (Pearson or Spearman), which are also commonly reported in the literature.

4.5. RESULTS AND DISCUSSION

In this section, we first compare the QSPR models with GNNs for the CC impact category. Next, we discuss the results of single- and multi-task GNN with additional country and energy mix features.

4.5.1. COMPARING QSPR AND GNN FOR THE CLIMATE CHANGE IMPACT CATEGORY

We benchmark our proposed models on the CC impact category, also as referred to GWP or GWI in the literature, because this impact category is widely studied in the literature. Table 4.5 shows results of the MRE and R^2 for this benchmark.

First, we compare the QSPR models to three GNN models. One observation is that GNN-M outperforms the QSPR model as MRE decreases by 6% and R^2 increases by 46.9%. This indicates that in this case, the end-to-end learning manner enables GNN-based models to automatically extract effective features from molecular graphs. Figure 4.5a further verifies that the GNN-M (blue) demonstrates improved predictive accuracy compared to the QSPR model (orange). GNN-M predictions are more tightly clustered along the parity line. Figure 4.4 compares the MRE of GNN-M and QSPR models across 761 molecules. The x-axis shows molecule indices sorted by GNN-M's error from lowest to highest, while the y-axis shows the MRE for each molecule. The plot illustrates how prediction accuracy varies per molecule and highlights that GNN-M generally achieves lower errors than QSPR. Notably, the highest prediction errors from GNN-M are primarily associated with sulfur-containing compounds. This is likely because sulfur-related chemicals often require detailed process-specific information—such as oxidation states, reaction pathways, or treatment technologies—which cannot be captured by molecular descriptors alone. To further compare the predictions of sulfur-related chemicals remains challenging, as, to the best of our knowledge, no existing studies report model predictions for specific individual molecules in this category. Moreover, the incorporation of the country and energy mix features improved both R^2 and RMSE performance, as depicted by GNN-C and GNN-E results. For example, compared with GNN-M, the R^2 increases by 21.8% and 9%, and RMSE drops by 28.7% and 23.8% for GNN-C and GNN-E, respectively. Additionally, both GNN-C and GNN-E models have lower standard deviations than GNN-M, which indicates that the results are likely more reproducible and reliable. Furthermore, the GNN-C model depicts slightly better performance than the GNN-E model in terms of MRE and R^2. One plausible reason is that the preprocessing of the energy mix data requires several simplifications due to the lack of information from several countries. Those simplifications may account for the high variance and less accurate predictions than GNN-E.

When inspecting Figure 4.5b, both GNN-C and GNN-E models are able to

capture the overall trend. However, substantial deviations from the parity line are observed across both models. The GNN-E model (purple) tends to overestimate the impact for low- to mid-range values, while the GNN-C model (green) behaves more conservatively, but underestimates in most cases. Notably, the sulfur dichloride process has a true CC impact greater than 45 in the plot. Both models significantly underestimate this value, predicting well below 10. This discrepancy arises because the majority of sulfur-containing compounds in the training data have much lower CC values—typically below 2. For instance, disulfur dichloride (S_2Cl_2), which is structurally similar to sulfur dichloride (SCl_2), consistently shows CC values around 1.5 across various countries. As a result, the models generalize a learned pattern that "sulfur-containing compounds" are associated with low CC, and fail to account for outliers like sulfur dichloride. This large gap is explained by differences in industrial production processes. Disulfur dichloride is synthesized directly from molten sulfur and chlorine gas in a relatively simple and energy-efficient process. In contrast, sulfur dichloride is produced by further chlorinating disulfur dichloride, requiring additional energy, materials, and process complexity [4]. This is a limit of molecular-based predictions when process context (e.g., feedstock origin, energy use, reaction severity) is crucial.

At first glance, the results in Table 4.5 show that our GNN-C and GNN-E models yield R^2 values that are comparable with, but in most cases lower than, those reported in the literature. Specifically, the GNN-C model achieves an R^2 of 0.39, which is comparable with the R^2 values of 0.37 and 0.36 reported by Wernet et al. [23] and Kleinekorte et al. [30], respectively. Furthermore, when compared with Karka et al. [31], GNN-C and GNN-E models demonstrate a substantially lower R^2, implying a potential need for integrating additional process information to enhance prediction accuracy.

However, it is important to note that our models exhibit higher MRE compared to previous studies, with MRE values ranging from 72% to 77%, whereas earlier works report values between 30% and 58.2%. These differences can be partially attributed to the distinct methodologies and datasets employed. Our study leverages a significantly larger dataset of 761 molecules, which is substantially greater than the 63 to 338 molecules used in previous studies. While this expanded dataset enhances the potential for more generalizable models, it also increases the complexity of the prediction task, potentially leading to the observed higher MRE values. It is important to note that our dataset is based on commercially available and transparent data sources, ensuring broader applicability. However, the increased dataset size also introduces greater structural and physicochemical variability, which inherently makes direct comparisons with smaller datasets more challenging.

Additionally, a direct comparison of R^2 values across studies is complicated

by the use of different evaluation metrics and model configurations. For instance, some studies used Pearson [24, 30] or Spearman[28, 29] R^2 metrics, while we employed the coefficient of determination. This variability in evaluation methods further complicates the comparison. However, the standard deviation of R^2 in our models is significantly lower than in previous studies, suggesting better reproducibility and model stability. Overall, despite these challenges, our results suggest that GNN-C and GNN-E models are promising tools for environmental impact prediction due to their improved performance in R^2 and RMSE over baseline models like GNN-M, as well as their lower standard deviations, indicating better reproducibility. Their ability to incorporate complex features, such as country and energy mix, enhances their applicability across different impact categories, potentially making them valuable for further development in environmental modeling.

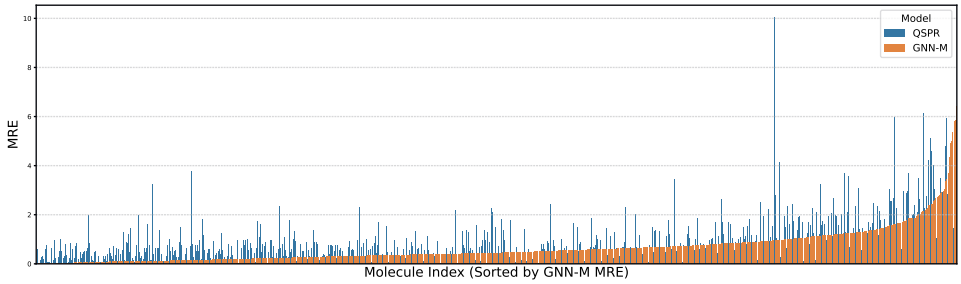
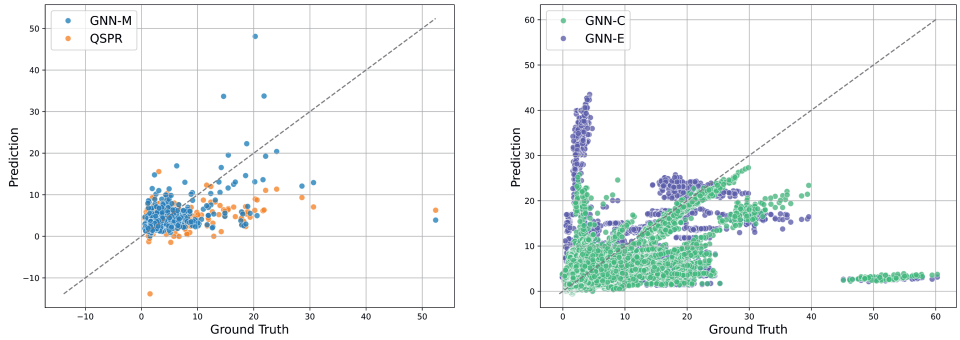


Figure 4.4: Per-molecule Mean Relative Error (MRE) comparison between the GNN-M and QSPR models. Molecules are sorted by GNN-M MRE to highlight performance differences across the dataset.



(a) Parity plot of QSPR and GNN-M models

(b) Parity plot of GNN-C and GNN-E models

Figure 4.5: Parity plots of QSPR, GNN-M, GNN-C, and GNN-E for predicting CC category.

Table 4.5: Results of QSPR and GNN models, as well as previous literature, for CC impact categories. The assessment metrics utilized include MRE and R^2 on the test dataset. Each metric represents the mean value ± standard deviation of the best models.

Model	Model type	Dataset (# molecules)	Additional features	MRE /%	R^2
QSPR	MLPs	761	-	108 ± 6	0.17 ± 0.03
GNN-M	GNN	761	-	101 ± 30	0.32 ± 0.09
GNN-C	GNN	761	✓	72 ± 5	0.39 ± 0.04
GNN-E	GNN	761	✓	77 ± 2	0.35 ± 0.02
Literature					
Wernet et al. [22]	MLPs & linear models	103	-	-	0.37 ± 0.36
Wernet et al. [23]	MLPs	338	-	58.2	0.41 ± 0.23
Song et al. [24]	MLPs	166	-	50	0.48[a]
Calvo-Serrano [27, 28]	MINLPs	83	✓	41.82	0.55[b]
Calvo-Serrano [29]	MINLPs	90	✓	30	0.55[b]
Kleinekorte et al. [30]	MLPs	63	✓	-	0.30[a]
Baxevanidis et al. [26]	Six linear/nonlinear models	214	-	53	0.26
Karka et al. [31]	MLPs & classification tree	-	✓	-	0.72 ± 0.14
Kleinekorte et al. [41]	GNN	166	✓	-	0.53
Zhang et al. [42]	GNN & Transformer	547	-	38.6	-

[a] R^2 Pearson
[b] R^2 Spearman

4.5.2. SINGLE-TASK MODEL GNN

Table 4.7 shows the performance of the single-task GNN-C and GNN-E models across fifteen impact categories in terms of MRE and R^2. As Section 4.5.1 discussed, here we omit training the QSPR and GNN-M because GNN-C and GNN-E depict more promising results.

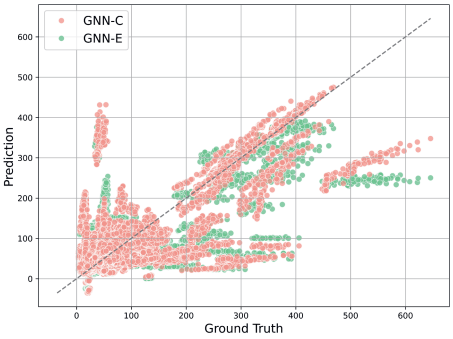
First, although both GNN-C and GNN-E models demonstrate some ability to predict the majority of impact categories, the overall performance is limited. For example, nine out of fifteen R^2 and ten out of fifteen R^2 are non-negative for GNN-C and GNN-E, respectively. Additionally, neither GNN-C nor GNN-E consistently outperforms the other across all categories. Specifically, GNN-E shows higher R^2 values in ten out of fifteen categories than GNN-C, suggesting that GNN-E may better explain the variability in the data for those categories. This could be important in cases where the focus is on understanding overall trends and patterns, such as exploratory analyses or when dealing with noisy data. On the other hand, GNN-C exhibits a lower MRE in twelve out of fifteen categories than GNN-E, which may indicate that GNN-C provides more accurate predictions with smaller average errors. This might be more relevant when precise predictions are needed, such as in applications where the magnitude of errors has a significant impact. As a result, the choice of model should be guided by the specific requirements of the impact category under study. For example, a model with a higher R^2 may be preferable for explaining variability, while a model with

lower MRE might be chosen when accuracy and precision are critical

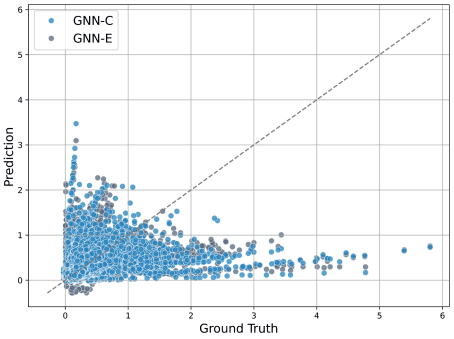
Moreover, both models show better performance in predicting CC, ER, EUF, and WU categories than others. This indicates that the additional country and energy mix features can account for geographical, regulatory, and energy-related differences across countries and provide models with more context for predicting those environmental impacts. For example, GNN-E has superior performance on water-related impact categories such as EUF and WU. One potential reason is that energy mix features provide related information about energy consumption patterns and their associated water footprints across different regions. Energy production and consumption significantly influence water usage, with various energy sources requiring differing amounts of water for extraction, processing, and generation, thereby giving more accurate predictions. However, it is important to note that the models exhibit higher prediction errors in certain categories, such as LU and IR. This is expected, as these impact categories are less directly influenced by molecular properties and are instead driven by broader land management practices and nuclear-related processes, which are not explicitly captured in our input features.

Furthermore, we selected the ER category and IR category to discuss the performance through Figure 4.6. In Figure 4.6a, both models demonstrate reasonable performance for the ER category, with the majority of predictions aligning with the parity line. However, we observe that one region (top left) is significantly overestimated, and one region (middle right) is underestimated. When inspecting the corresponding chemicals, all significant prediction errors—both underestimations (e.g., carbon disulfide, 1,3,4-thiadiazolidine-2,5-dithione, and tetramethyl thiuram disulfide) and overestimations (e.g., thioformamide)—originated from sulfur-containing compounds. As shown in Figure 4.7a, sulfur-related chemicals exhibit a highly skewed and widespread ER distribution, ranging from very low (< 10) to extremely high values (> 600). In contrast, the ER values for alkene-containing molecules (Figure 4.7b)) form a narrower (10 - 90) and more symmetric distribution centered around moderate values. The wide variance in ER values for sulfur-related compounds likely limits the model's generalizability. This may stem from the diverse nature of sulfur production processes, which differ substantially in energy intensity depending on feedstocks, technologies, and purification steps. These results highlight the need to consider intra-family heterogeneity and suggest that incorporating process descriptors may improve prediction for such chemically diverse families. On the other hand, Figure 4.6b reveals that both models significantly underperform on the IR category. Predictions for both GNN-C and GNN-E are heavily clustered at low values and fail to reproduce the tail of the ground truth distribution.

In conclusion, Table 4.7 suggests that neither model consistently outperforms

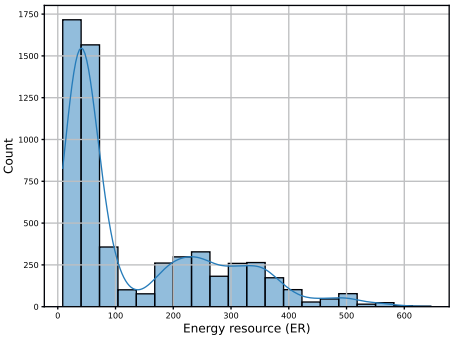


(a) Parity plot of single-task GNN-C and GNN-E models for ER.

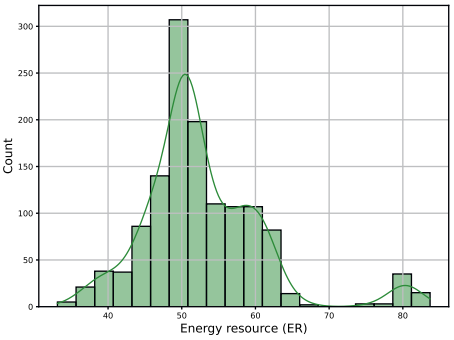


(b) Parity plot of single-task GNN-C and GNN-E models for IR.

Figure 4.6: Parity plots of single-task GNN-C and GNN-E for ER and IR.



(a) Data distribution of sulfur-related family (ER).



(b) Data distribution of alkenes-related family (ER).

Figure 4.7: Data distributions for sulfur- and alkenes-related families (ER).

the other across all categories. The choice between GNN-C and GNN-E may depend on the specific impact category of interest. Therefore, we further deploy multi-task GNN to investigate the causes of high variance in model predictions, assess the robustness of the models, and explore potential improvements in model architecture or training data.

Table 4.6: Results of single-task GNN-C and GNN-E models evaluated across fifteen categories. The assessment metrics utilized include MRE and R^2. Each metric represents the mean value ± standard deviation obtained from three independent runs.

Impact	Single-task GNN-C		Single-task GNN-E	
category	MRE/%	R^2	MRE/%	R^2
AC	**98 ± 8**	-0.03 ± 0.02	122 ± 14	**0.04 ± 0.08**
CC	**72 ± 5**	**0.39 ± 0.04**	77 ± 2	0.35 ± 0.02
ECO	**193 ± 16**	**0.17 ± 0.08**	232 ± 30	0.16 ± 0.09
ER	**159 ± 2**	0.47 ± 0.04	166 ± 10	**0.50 ± 0.05**
EUF	**195 ± 48**	0.40 ± 0.16	239 ± 15	**0.58 ± 0.06**
EUM	87 ± 8	-0.01 ± 0.01	99 ± 5	**0.01 ± 0.01**
EUT	92 ± 6	-0.02 ± 0.03	**88 ± 3**	**-0.01 ± 0.01**
HT	413 ± 71	-0.18 ± 0.03	**332 ± 247**	**-0.14 ± 0.2**
IR	**589 ± 32**	**-0.17 ± 0.11**	670 ± 20	-0.27 ± 0.08
LU	**253 ± 10**	-0.04 ± 0.02	272 ± 36	**-0.01 ± 0.03**
MR	1405 ± 1312	0.16 ± 0.35	**580 ± 109**	**0.32 ± 0.37**
OD	**414 ± 22**	**0.09 ± 0.09**	447 ± 11	-0.14 ± 0.06
PMF	**176 ± 4**	0.12 ± 0.04	281 ± 28	**0.13 ± 0.03**
POF	**85 ± 4**	0.004 ± 0.022	89 ± 6	**0.12 ± 0.06**
WU	**237 ± 42**	**0.36 ± 0.19**	242 ± 12	0.33 ± 0.10

4.5.3. MULTI-TASK GNN

We further investigate multi-task GNN-C and GNN-E models and the MRE and R^2 across fifteen impact categories (see results in Table 4.7). Furthermore, Figure 4.9 provides a clear comparison of the performance between single-task and multi-task GNN models.

First, we observe a consistent improvement in MRE across nearly all impact categories when compared to the single-task GNN-C and GNN-E as the Figure 4.9a shows. For example, the MRE of fourteen out of fifteen categories for multi-task GNN-C decreases by ranging from 8% (PMF category) to 73% (OD category) compared with single-task GNN-C. The same trend echoes in multi-task GNN-E. The MRE of thirteen out of fifteen categories decreases by ranging from 16% (CC category) to 79% (IR category) compared with single-task counterparts. This enhancement verifies the hypothesis that multi-task GNN models learn a shared representation that captures underlying patterns common to all tasks, thereby building a more robust and generalizable model framework. This shared learning facilitates the transfer of knowledge between tasks, with insights or patterns learned in one task potentially benefiting others, particularly if they are interrelated.

For multi-task GNN models, eleven out of fifteen R^2 are non-negative for both

GNN-C and GNN-E models. When compared with their single-task counterparts, the R^2 presents a mixed landscape with some categories showing improvements and others slight declines as Figure 4.9b depicts. For instance, for multi-task GNN-C, in the categories of AC, HT, and IR, the performance has been improved by 115%, 152%, and 185%. While for categories ECO and POF, the R^2 significantly decreases by 216% and 112%. Additionally, for multi-task GNN-E, we do observe significant improvements in HT (131%), IR (456%), and LU (327%) categories, but for the ECO, EUM, and POF categories, the R^2 decreases by 132%, 224% and 260%, respectively, compared with single-task GNN-E. Notably, for both GNN-C and GNN-E, the multi-task models significantly improve the R^2 in HT and IR categories but decline in ECO and POF categories. This discrepancy could potentially be attributed to the nature of the tasks and their underlying data structures. HT and IR categories share similarities in how environmental exposures impact human health, allowing the multi-task GNN to leverage shared representations and improve predictive performance. In contrast, ECO and POF involve complex, distinct mechanisms—ecological interactions in aquatic environments for ECO, and atmospheric chemistry for POF—that do not easily align with the patterns learned from other tasks. Consequently, the multi-task model may struggle to generalize across these divergent categories, leading to a worse performance. These findings highlight the importance of task similarity and data alignment in multi-task learning frameworks, suggesting that while multi-task GNNs can enhance performance for related tasks, they may require careful design or even task-specific models when dealing with highly specialized or unrelated properties.

We further examine the parity plots for ER and IR categories for multi-task models. Compared to the single-task results, both models exhibit slightly improved alignment with the parity line, particularly in the mid-range values of the ER category (Figure 4.8a). The overall trend is supported by the performance metrics: the corresponding improvements in R^2 and reductions in MRE (as shown in Table 4.7) confirm that generalization and error mitigation have indeed improved. For the IR category (Figure 4.8b), although predictions still remain challenging, both models show improved dispersion compared to their single-task counterparts.

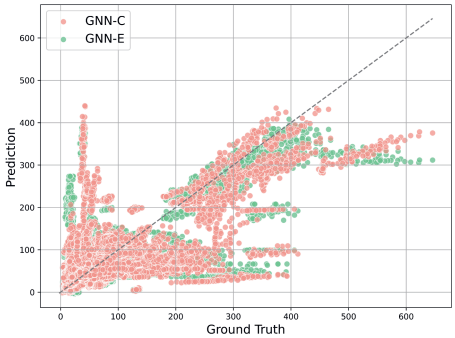
For the CC category, multi-task GNN models perform worse than single-task models. This may be because multi-task learning prioritizes general features, reducing the ability of the model to capture CC-specific patterns. Therefore, a single-task GNN is recommended for predicting CC category. Additionally, to cross-compare the multi-task results of GNN-C and GNN-E, both models display similar performances in terms of MRE and R^2. GNN-E marginally outperforms in eight impact categories for R^2 and maintains overall smaller standard deviations compared to GNN-C, indicating more reproducible results. On the other hand,

GNN-C shows better MRE in ten out of fifteen impact categories than GNN-E.

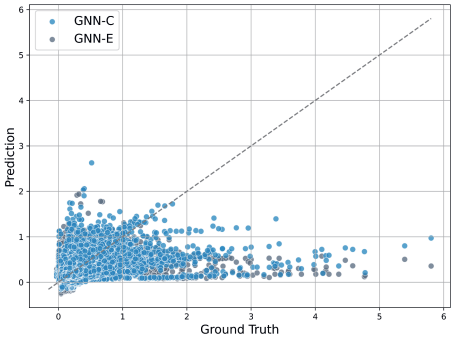
Additionally, Figure 4.10 shows the mean relative error (MRE, log scale) for propylene predicted by the multitask GNN-C and GNN-E models across the fifteen EF midpoint impact categories. Errors span more than three orders of magnitude: the lowest MREs occur for ER, OD, and IR (all below roughly the 20% range), whereas MR and WU exhibit extremely large errors (>10^3%), with EUf and EUm also high. Intermediate error levels (tens to low hundreds of percent) are observed for AC, CC, EUt, LU, PMF, and POF. Categories such as ER, OD, and IR are largely governed by upstream energy-conversion or halocarbon-related emission factors that scale well with the elemental composition of propylene and the national energy mix. They also show comparatively low inter-database variability, which helps the model keep MREs below 20 % [69, 70]. Conversely, MR and WU require detailed information on ore grades, mining routes, and regional water scarcity indices, all of which vary widely across countries and databases [69, 70]. Because such process- and location-specific flows are absent from the molecular graph, the model under-predicts or over-predicts by orders of magnitude. Comparing the two models, adding energy-mix information (GNN-E, orange) substantially improves ECO and modestly reduces LU (and slightly ER/IR), but increases error relative to the country-only variant (GNN-C, blue) in categories such as AC, EUm, HT, MR, PMF, POF, and WU. Taken together, the figure highlights that molecular-based multitask models are most useful for energy- or chemistry-dominated categories, whereas process-intensive categories will still require additional life-cycle inventory data for reliable screening. This illustrative case also underscores the limitations of predictive models in practical applications—particularly in later stages of process development, where accurate life-cycle assessments depend on detailed process designs and simulation studies beyond molecular structure and national energy mix.

Overall, we offer the following recommendations for future work in early-stage prediction of life cycle impact categories. First, the choice of impact category is critical. When using only molecular features in combination with geographical or energy mix data, categories such as land use (LU) may be less suitable due to their weak correlation with molecular structure. We recommend excluding such categories from purely structure-based modeling efforts. Second, it is important to inspect the data distribution, particularly with respect to chemical diversity and class imbalance. If a small subset of molecules dominates the dataset or if certain classes are underrepresented, data augmentation or rebalancing strategies [71] could be considered to improve model generalization in future work. Finally, for applications involving multiple impact categories, we suggest initially developing a multi-task GNN model using molecular and energy mix or geographical features. This approach enables shared representation learning across tasks. If perfor-

mance is suboptimal for certain categories, those specific targets may benefit from additional fine-tuning using task-specific (single-task) GNNs. These strategies can help improve the robustness and interpretability of predictive models in the context of sustainable process and product design.



(a) Parity plot of multi-task GNN-C and GNN-E models for ER.



(b) Parity plot of multi-task GNN-C and GNN-E models for IR.

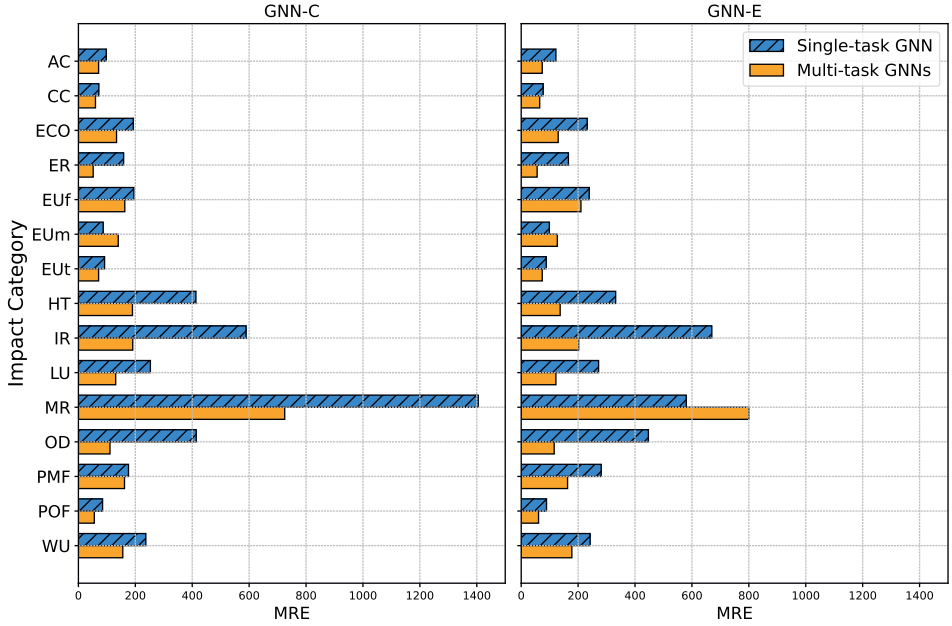
Figure 4.8: Parity plots of multi-task GNN-C and GNN-E for ER and IR.

Table 4.7: Results of multi-task GNN-C and GNN-E models evaluated across fifteen categories. The assessment metrics utilized include MAE, MRE, and R^2. Each metric represents the mean value ± standard deviation obtained from three independent runs.

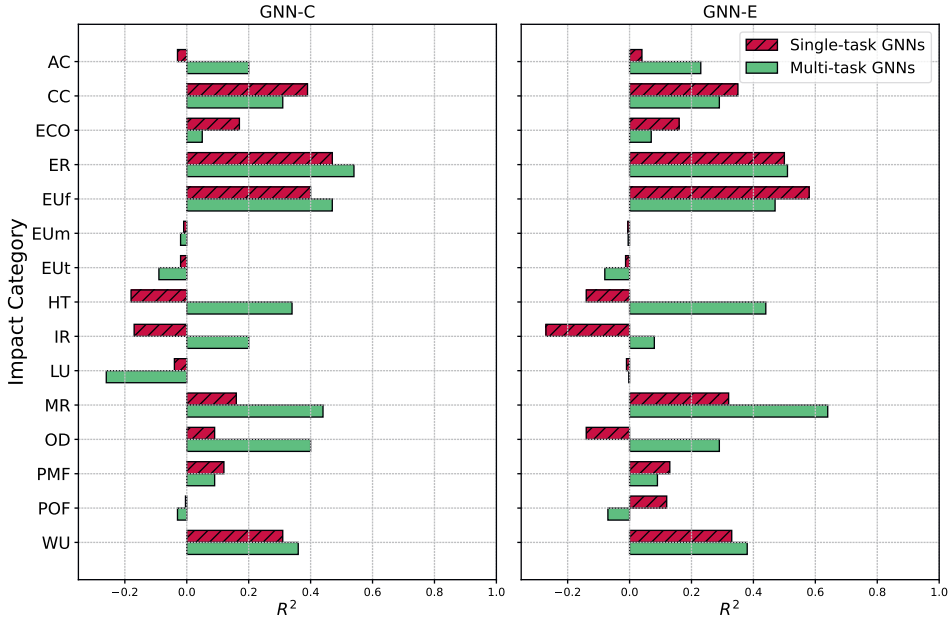
Impact	Multi-task GNN-C		Multi-task GNN-E	
category	MRE/%	R^2	MRE/%	R^2
AC	71 ± 1	0.20 ± 0.09	74 ± 1	**0.23 ± 0.04**
CC	60 ± 2	**0.31 ± 0.04**	65 ± 3	0.29 ± 0.03
ECO	134 ± 9	0.05 ± 0.09	130 ± 8	**0.07 ± 0.02**
ER	52 ± 1	**0.54 ± 0.03**	56 ± 1	0.51 ± 0.02
EUF	163 ± 18	0.47 ± 0.10	210 ± 27	**0.47 ± 0.02**
EUM	140 ± 27	-0.02 ± 0.00	127 ± 4	**-0.005 ± 0.006**
EUT	71 ± 2	-0.09 ± 0.07	74 ± 6	**-0.08 ± 0.09**
HT	190 ± 22	0.34 ± 0.24	137 ± 16	**0.44 ± 0.02**
IR	191 ± 6	**0.20 ± 0.01**	202 ± 3	0.08 ± 0.00
LU	131 ± 7	-0.26 ± 0.45	122 ± 11	**-0.002 ± 0.037**
MR	725 ± 219	0.44 ± 0.46	800 ± 189	**0.64 ± 0.11**
OD	111 ± 3	**0.40 ± 0.05**	116 ± 0	0.29 ± 0.02
PMF	162 ± 7	0.09 ± 0.04	163 ± 4	**0.09 ± 0.03**
POF	56 ± 1	**-0.03 ± 0.00**	61 ± 3	-0.07 ± 0.01
WU	156 ± 15	0.36 ± 0.19	178 ± 7	**0.38 ± 0.05**

4.6. CONCLUSION

We propose an end-to-end GNN-based approach for predicting fifteen distinct environmental impact categories using a comprehensive dataset from CarbonMind. This dataset includes 51,905 processes producing 761 unique molecules across 91 countries. We further integrate energy mix data from the IEA, corresponding to



(a) MRE results for GNN-C and GNN-E under single-task and multi-task scenarios.



(b) R^2 results for GNN-C and GNN-E under single-task and multi-task scenarios.

Figure 4.9: MRE and R^2 results for GNN-C and GNN-E under single- and multi-task scenarios.

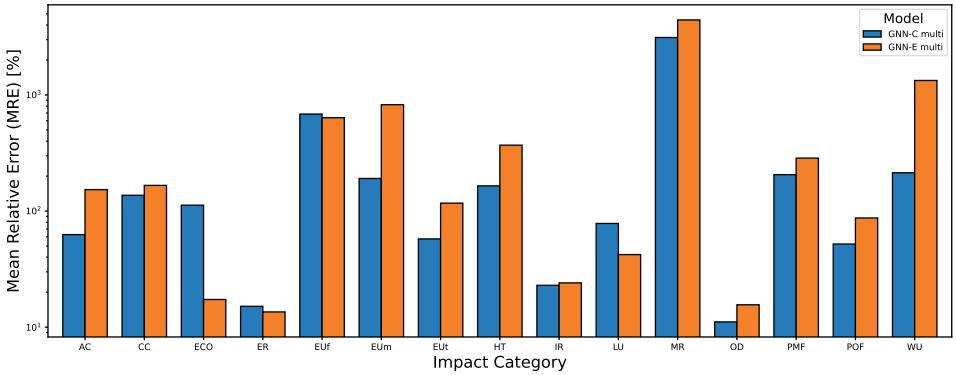


Figure 4.10: Mean relative error (MRE, log scale) for propylene predicted with the multitask GNN-C (blue) and GNN-E (orange) models across 15 impact categories. For each category, the bar represents the MRE averaged over all country contexts, while the GNN-E bar is averaged over the corresponding national energy-mix scenarios.

the same countries, to enrich our dataset. Our analysis begins by comparing the performance of QSPR and GNN models specifically for the climate change impact category, revealing that GNN models outperform the QSPR model. Furthermore, benchmarking our GNN models against existing literature for the climate change category demonstrates that our models achieve comparable performance. Expanding our approach, we developed both single- and multi-task GNN-C and GNN-E models to predict all fifteen impact categories. The results suggest that multi-task learning can enhance model performance in complex environmental impact predictions compared to single-task GNNs, considering the MRE. In summary, we recommend selecting impact categories based on their relevance to molecular structure, and excluding those with weak correlation—such as land use (LU)—when relying solely on molecular and contextual features. It is also important to examine the data distribution and address class imbalance to improve training performance. For multi-category prediction tasks, we suggest first training a multi-task GNN with molecular and geographic or energy mix features. If certain categories show poor predictive performance, dedicated single-task models can then be developed for those specific targets. Overall, the predictive performance of our computational models for impact categories remains limited. In scenarios where detailed process data is unavailable or inaccessible - such as early-stage process design, supply chain assessments, or cases involving proprietary or confidential information - the models can provide useful approximations to support preliminary decision-making.

4.7. ACKNOWLEDGEMENTS

This work was supported by the Netherlands Organisation for Scientific Research (NWO) under the Veni grant.

4.8. DECLARATION OF GENERATIVE AI AND AI-ASSISTED TECHNOLOGIES IN THE WRITING PROCESS

During the preparation of this work the author(s) used ChatGPT 4o in order to correct the grammar and polish the sentences. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the published article.

REFERENCES

[1] G. A. Buchner, K. J. Stepputat, A. W. Zimmermann, and R. Schomäcker, *Specifying technology readiness levels for the chemical industry,* Industrial & Engineering Chemistry Research **58**, 6957 (2019).

[2] J. M. Weber, Z. Guo, C. Zhang, A. M. Schweidtmann, and A. A. Lapkin, *Chemical data intelligence for sustainable chemistry,* Chemical Society Reviews **50**, 12013 (2021).

[3] J. M. Weber, Z. Guo, and A. A. Lapkin, *Discovering circular process solutions through automated reaction network optimization,* ACS Engineering Au **2**, 333 (2022).

[4] K. Ulonska, M. Skiborowski, A. Mitsos, and J. Viell, *Early-stage evaluation of biorefinery processing pathways using process network flux analysis,* AIChE Journal **62**, 3096 (2016).

[5] H. Minten, B. D. Vandegehuchte, B. Jaumard, R. Meys, C. Reinert, and A. Bardow, *Early-stage impact assessment tool (estimate) for the life cycle assessment of co2-based chemicals,* Green Chemistry **26**, 8728 (2024).

[6] N. Preuss, A. S. Alshehri, and F. You, *Large language models for life cycle assessments: Opportunities, challenges, and risks,* Journal of Cleaner Production **466**, 142824 (2024).

[7] C. Blanco, N. Pauliks, F. Donati, N. Engberg, and J. Weber, *Machine learning to support prospective life cycle assessment of emerging chemical technologies,* Current Opinion in Green and Sustainable Chemistry **50**, 100979 (2024).

[8] I. Standard, *Environmental management-life cycle assessment-requirements and guidelines,* London: ISO (2006).

[9] M. Heidari, D. Mathis, P. Blanchet, and B. Amor, *Streamlined life cycle assessment of an innovative bio-based material in construction: A case study of a phase change material panel,* Forests **10**, 160 (2019).

[10] S. Nakamura and K. Nansai, *Input–output and hybrid lca,* in *Special Types of Life Cycle Assessment* (Springer Netherlands, 2016) pp. 219–291.

[11] T. J. Mattila, *Use of input–output analysis in lca,* in *Life Cycle Assessment* (Springer International Publishing, 2017) pp. 349–372.

[12] Y. Yang, R. Heijungs, and M. Brandão, *Hybrid life cycle assessment (lca) does not necessarily yield more accurate results than process-based lca,* Journal of Cleaner Production **150**, 237 (2017).

[13] A. Martínez-Rocamora, J. Solís-Guzmán, and M. Marrero, *Lca databases focused on construction materials: A review,* Renewable and Sustainable Energy Reviews **58**, 565 (2016).

[14] G. Wernet, C. Bauer, B. Steubing, J. Reinhard, E. Moreno-Ruiz, and B. Weidema, *The ecoinvent database version 3 (part i): overview and methodology,* The International Journal of Life Cycle Assessment **21**, 1218 (2016).

[15] S. Fazio, D. Garraín, F. Mathieux, C. De la Rúa, M. Recchioni, and Y. Lechón, *Method applied to the background analysis of energy data to be considered for the european reference life cycle database (elcd),* SpringerPlus **4** (2015), 10.1186/s40064-015-0914-x.

[16] J. Kleinekorte, L. Fleitmann, M. Bachmann, A. Kätelhön, A. Barbosa-Póvoa, N. von der Assen, and A. Bardow, *Life cycle assessment for the design of chemical processes, products, and supply chains,* Annual Review of Chemical and Biomolecular Engineering **11**, 203 (2020).

[17] A. G. Parvatker and M. J. Eckelman, *Comparative evaluation of chemical life cycle inventory generation methods and implications for life cycle assessment results,* ACS Sustainable Chemistry & Engineering **7**, 350 (2018).

[18] A. S. Alshehri, A. K. Tula, F. You, and R. Gani, *Next generation pure component property estimation models: With and without machine learning techniques,* AIChE Journal **68** (2021), 10.1002/aic.17469.

[19] G. Chen, Z. Song, Z. Qi, and K. Sundmacher, *A scalable and integrated machine learning framework for molecular properties prediction,* AIChE Journal **69** (2023), 10.1002/aic.18185.

[20] L. P. Hammett, *Some relations between reaction rates and equilibrium constants.* Chemical Reviews **17**, 125 (1935).

[21] R. Gani, *Group contribution-based property estimation methods: advances and perspectives,* Current Opinion in Chemical Engineering **23**, 184 (2019).

[22] G. Wernet, S. Hellweg, U. Fischer, S. Papadokonstantakis, and K. Hungerbühler, *Molecular-structure-based models of chemical inventories using neural networks,* Environmental Science & Technology **42**, 6717 (2008).

[23] G. Wernet, S. Papadokonstantakis, S. Hellweg, and K. Hungerbühler, *Bridging data gaps in environmental assessments: Modeling impacts of fine and basic chemical production,* Green Chemistry **11**, 1826 (2009).

[24] R. Song, A. A. Keller, and S. Suh, *Rapid life-cycle impact screening using artificial neural networks,* Environmental Science & Technology **51**, 10777 (2017).

[25] Y. Sun, X. Wang, N. Ren, Y. Liu, and S. You, *Improved machine learning models by data processing for predicting life-cycle environmental impacts of chemicals,* Environmental Science & Technology **57**, 3434 (2022).

[26] P. Baxevanidis, S. Papadokonstantakis, A. Kokossis, and E. Marcoulaki, *Group contribution-based lca models to enable screening for environmentally benign novel chemicals in camd applications,* AIChE Journal **68** (2021), 10.1002/aic.17544.

[27] R. Calvo-Serrano, M. González-Miquel, S. Papadokonstantakis, and G. Guillén Gosálbez, *Cradle-to-gate environmental impact prediction from chemical attributes using mixed-integer programming,* in *27th European Symposium on Computer Aided Process Engineering* (Elsevier, 2017) pp. 1999–2004.

[28] R. Calvo-Serrano, M. González-Miquel, S. Papadokonstantakis, and G. Guillén-Gosálbez, *Predicting the cradle-to-gate environmental impact of chemicals from molecular descriptors and thermodynamic properties via mixed-integer programming,* Computers & Chemical Engineering **108**, 179 (2018).

[29] R. Calvo-Serrano, M. González-Miquel, and G. Guillén-Gosálbez, *Integrating cosmo-based σ-profiles with molecular and thermodynamic attributes to predict the life cycle environmental impact of chemicals,* ACS Sustainable Chemistry & Engineering **7**, 3575 (2018).

[30] J. Kleinekorte, L. Kröger, K. Leonhard, and A. Bardow, *A neural network-based framework to predict process-specific environmental impacts,* in *29th European Symposium on Computer Aided Process Engineering* (Elsevier, 2019) pp. 1447–1452.

[31] P. Karka, S. Papadokonstantakis, and A. Kokossis, *Digitizing sustainable process development: From ex-post to ex-ante lca using machine-learning to evaluate bio-based process technologies ahead of detailed design,* Chemical Engineering Science **250**, 117339 (2022).

[32] A. Sultan, M. Rausch-Dupont, S. Khan, O. Kalinina, A. Volkamer, and D. Klakow, *Transformers for molecular property prediction: Domain adaptation efficiently improves performance,* (2025), 10.48550/ARXIV.2503.03360, arXiv:2503.03360 [cs.LG] .

[33] A. Sultan, J. Sieg, M. Mathea, and A. Volkamer, *Transformers for molecular property prediction: Lessons learned from the past five years,* (2024), 10.48550/ARXIV.2404.03969, arXiv:2404.03969 [cs.LG] .

[34] S. Zhang, Y. Liu, and L. Xie, *Molecular mechanics-driven graph neural network with multiplex graph for molecular structures,* (2020), 10.48550/ARXIV.2011.07457, arXiv:2011.07457 [cs.LG] .

[35] D. Buterez, J. P. Janet, S. J. Kiddle, D. Oglic, and P. Lió, *Transfer learning with graph neural networks for improved molecular property prediction in the multi-fidelity setting,* Nature Communications **15** (2024), 10.1038/s41467-024-45566-8.

[36] H. Wang, A. Zhang, Y. Zhong, J. Tang, K. Zhang, and P. Li, *Chain-aware graph neural networks for molecular property prediction,* Bioinformatics **40** (2024), 10.1093/bioinformatics/btae574.

[37] Q. Gao, T. Dukker, A. M. Schweidtmann, and J. M. Weber, *Self-supervised graph neural networks for polymer property prediction,* Molecular Systems Design & Engineering **9**, 1130 (2024).

[38] D. Trivedi, K. Patrikar, and A. Mondal, *Graph-based networks for accurate prediction of ground and excited state molecular properties from minimal features,* Molecular Systems Design & Engineering **9**, 1275 (2024).

[39] J. G. Rittig, Q. Gao, M. Dahmen, A. Mitsos, and A. M. Schweidtmann, *Graph neural networks for the prediction of molecular structure-property relationships,* (2022), 10.48550/ARXIV.2208.04852, arXiv:2208.04852 [q-bio.BM] .

[40] P. Pope, S. Kolouri, M. Rostrami, C. Martin, and H. Hoffmann, *Discovering molecular functional groups using graph convolutional neural networks,* (2018), 10.48550/ARXIV.1812.00265, arXiv:1812.00265 [cs.LG] .

[41] J. Kleinekorte, J. Kleppich, L. Fleitmann, V. Beckert, L. Blodau, and A. Bardow, *Appropriate life cycle assessment: A process-specific, predictive impact assessment method for emerging chemical processes,* ACS Sustainable Chemistry & Engineering **11**, 9303 (2023).

[42] D. Zhang, Z. Wang, C. Oberschelp, E. Bradford, and S. Hellweg, *Enhanced deep-learning model for carbon footprints of chemicals,* ACS Sustainable Chemistry & Engineering **12**, 2700 (2024).

[43] R. Liu, Y. Liu, J. Duan, F. Hou, L. Wang, X. Zhang, and G. Li, *Ensemble learning directed classification and regression of hydrocarbon fuels,* Fuel **324**, 124520 (2022).

[44] B. Ramsundar, S. Kearnes, P. Riley, D. Webster, D. Konerding, and V. Pande, *Massively multitask networks for drug discovery,* (2015), 10.48550/ARXIV.1502.02072, arXiv:1502.02072 [stat.ML] .

[45] X. Chen, Z.-J. Zhang, X. Hong, and L. Ackermann, *Integrating a multitask graph neural network with dft calculations for site-selectivity prediction of arenes and mechanistic knowledge generation,* Nature Synthesis (2025), 10.1038/s44160-025-00770-2.

[46] F. Hu, D. Chen, Q. Liu, and S. Wu, *Improving multi-task gnns for molecular property prediction via missing label imputation,* Machine Intelligence Research **22**, 131 (2025).

[47] European Commission, *Commission recommendation (eu) 2021/2279 of 15 december 2021 on the use of the environmental footprint methods to measure and communicate the life cycle environmental performance of products and organisations,* `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32021H2279` (2021), official Journal of the European Union, L 471, 30.12.2021, p. 1–396.

[48] A. R. Katritzky, M. Kuanar, S. Slavov, C. D. Hall, M. Karelson, I. Kahn, and D. A. Dobchev, *Quantitative correlation of physical and chemical properties with chemical structure: Utility for prediction,* Chemical Reviews **110**, 5714 (2010).

[49] S. W. Benson, F. R. Cruickshank, D. M. Golden, G. R. Haugen, H. E. O'Neal, A. S. Rodgers, R. Shaw, and R. Walsh, *Additivity rules for the estimation of thermochemical properties,* Chemical Reviews **69**, 279 (1969).

[50] K. JOBACK and R. REID, *Estimation of pure-component properties from group-contributions,* Chemical Engineering Communications **57**, 233 (1987).

[51] R. Todeschini and V. Consonni, *Handbook of Molecular Descriptors* (Wiley, 2000).

[52] A. Cherkasov, E. N. Muratov, D. Fourches, A. Varnek, I. I. Baskin, M. Cronin, J. Dearden, P. Gramatica, Y. C. Martin, R. Todeschini, V. Consonni, V. E. Kuz'min, R. Cramer, R. Benigni, C. Yang, J. Rathman, L. Terfloth, J. Gasteiger, A. Richard, and A. Tropsha, *Qsar modeling: Where have you been? where are you going to?* Journal of Medicinal Chemistry **57**, 4977 (2014).

[53] A. M. Schweidtmann, J. G. Rittig, A. König, M. Grohe, A. Mitsos, and M. Dahmen, *Graph neural networks for prediction of fuel ignition quality,* Energy & Fuels **34**, 11395 (2020).

[54] A. M. Schweidtmann, J. G. Rittig, J. M. Weber, M. Grohe, M. Dahmen, K. Leonhard, and A. Mitsos, *Physical pooling functions in graph neural networks for molecular property prediction,* Computers & Chemical Engineering **172**, 108202 (2023).

[55] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, *Neural message passing for quantum chemistry,* (2017), 10.48550/ARXIV.1704.01212, arXiv:1704.01212 [cs.LG] .

[56] W. Hamilton, Z. Ying, and J. Leskovec, *Inductive representation learning on large graphs,* in *Advances in Neural Information Processing Systems*, Vol. 30, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, Inc., 2017).

[57] D. Weininger, *Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules,* Journal of Chemical Information and Computer Sciences **28**, 31 (1988).

[58] L. Stellner, A. Kätehön, O. Vögler, R. Hermanns, S. Suh, A. Bardow, and R. Meys, *Methodology cm.chemicals,* Carbon Minds GmbH, Cologne (2022).

[59] M. A. J. Huijbregts, Z. J. N. Steinmann, P. M. F. Elshout, G. Stam, F. Verones, M. Vieira, M. Zijp, A. Hollander, and R. van Zelm, *Recipe2016: a harmonised life cycle impact assessment method at midpoint and endpoint level,* The International Journal of Life Cycle Assessment **22**, 138 (2016).

[60] A. Mauri, V. Consonni, M. Pavan, R. Todeschini, *et al.*, *Dragon software: An easy approach to molecular descriptor calculations,* Match **56**, 237 (2006).

[61] G. Landrum, P. Tosco, B. Kelley, Ric, D. Cosgrove, Sriniker, R. Vianello, Gedeck, N. Schneider, G. Jones, E. Kawashima, D. Nealschneider, A. Dalke, B. Cole, M. Swain, S. Turk, A. Savelev, A. Vaucher, M. Wójcikowski, I. Take, V. F. Scalfani, D. Probst, K. Ujihara, R. Walker, G. Godin, A. Pahl, J. Lehtivarjo, F. Berenger, Strets123, and J. D. Biggs, *rdkit/rdkit: 2023_09_6 (q3 2023) release,* (2024), 10.5281/zenodo.591637.

[62] J. G. Rittig, K. Ben Hicham, A. M. Schweidtmann, M. Dahmen, and A. Mitsos, *Graph neural networks for temperature-dependent activity coefficient prediction of solutes in ionic liquids,* Computers & Chemical Engineering **171**, 108153 (2023).

[63] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, *Learning phrase representations using rnn encoder-decoder for statistical machine translation,* (2014), 10.48550/ARXIV.1406.1078, arXiv:1406.1078 [cs.CL] .

[64] M. Withnall, E. Lindelöf, O. Engkvist, and H. Chen, *Building attention and edge message passing neural networks for bioactivity and physical–chemical property prediction,* Journal of Cheminformatics **12** (2020), 10.1186/s13321-019-0407-y.

[65] M. Meng, Z. Wei, Z. Li, M. Jiang, and Y. Bian, *Property prediction of molecules in graph convolutional neural network expansion,* in *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*, IEEE (IEEE, 2019).

[66] S. Ruder, *An overview of multi-task learning in deep neural networks,* (2017), 10.48550/ARXIV.1706.05098, arXiv:1706.05098 [cs.LG] .

[67] P. Burkardt, M. Fleischmann, T. Wegmann, M. Braun, J. Knöll, L. Schumacher, F. vom Lehn, B. Lehrheuer, M. Meinke, H. Pitsch, R. Kneer, W. Schröder, and S. Pischinger, *On the use of active pre-chambers and bio-hybrid fuels in internal combustion engines,* in *Engines and Fuels for Future Transport* (Springer Singapore, 2021) pp. 205–231.

[68] G. E. Dahl, N. Jaitly, and R. Salakhutdinov, *Multi-task neural networks for qsar predictions,* (2014), 10.48550/ARXIV.1406.1231, arXiv:1406.1231 [stat.ML] .

[69] T. Kim, P. T. Benavides, J. D. Kneifel, K. L. Beers, and T. R. Hawkins, *Cross-database comparisons on the greenhouse gas emissions, water consumption, and fossil-fuel use of plastic resin production and their post-use phase impacts,* Resources, Conservation and Recycling **198**, 107168 (2023).

[70] I. Alyaseri and J. Zhou, *Handling uncertainties inherited in life cycle inventory and life cycle impact assessment method for improved life cycle assessment of wastewater sludge treatment,* Heliyon **5**, e02793 (2019).

[71] O. Wieder, S. Kohlbacher, M. Kuenemann, A. Garon, P. Ducrot, T. Seidel, and T. Langer, *A compact review of molecular property prediction with graph neural networks,* Drug Discovery Today: Technologies **37**, 1 (2020).

4.9. APPENDIX

Table 4.8: Overview of the energy mix data from IEA.

Product	Description
Coal, peat and oil shale	Coal includes all coal, both primary (including hard coal and lignite) and derived fuels (including patent fuel, coke oven coke, gas coke, BKB, gas works gas, coke oven gas, blast furnace gas and other recovered gases). Peat (including peat products) and oil shale are also included in this figure where applicable.
Crude, NGL and feedstocks	Crude oil comprises crude oil, natural gas liquids, refinery feedstocks and additives as well as other hydrocarbons (including emulsified oils, synthetic crude oil, mineral oils extracted from bituminous minerals such as oil shale, bituminous sand, etc. and oils from coal and gas liquefaction).
Oil products	Oil products comprise refinery gas, ethane, LPG, aviation gasoline, motor gasoline, jet fuels, kerosene, gas/diesel oil, fuel oil, naphtha, white spirit, lubricants, bitumen, paraffin waxes, petroleum coke and other oil products.
Natural gas	Natural gas includes both 'associated' and 'non-associated' gas as well as colliery gas (excluding natural gas liquids).
Nuclear	Nuclear shows the primary heat equivalent of the electricity produced by a nuclear power plant with an average thermal efficiency of 33%.
Renewables and waste	Renewables and waste comprises hydro, geothermal, solar, wind and tide/wave/ocean energy and the use of these energy forms for electricity and heat generation, as well as solid biofuels, liquid biofuels, biogases, industrial waste and municipal waste.
Electricity	Electricity shows final consumption and trade in electricity, which is accounted at the same heat value as electricity in final consumption (that is, 1 GWh = 0.0036 PJ).
Heat	Heat shows the disposition of heat produced for sale. The large majority of the heat included in this column results from the combustion of fuels although some small amounts are produced from electrically powered heat pumps and boilers. Any heat extracted from ambient air by heat pumps is shown as production.
Total	Total equals the total of products included in the dataset.

Table 4.9: 56 molecular descriptors from [24] used in QSPR models.

Descriptors abbreviation	Descriptors full name	Descriptor category
MW	Molecular weight	Constitutional indices
AMW	Average molecular weight	Constitutional indices
nBM	Number of multiple bonds	Constitutional indices
RBN	Number of rotatable bonds	Constitutional indices
nF	Number of Fluorine atoms	Constitutional indices
N%	Percentage of N atoms	Constitutional indices
O%	Percentage of O atoms	Constitutional indices
D/Dtr05	Distance/detour ring index of order 5	Ring descriptors
D/Dtr10	Distance/detour ring index of order 10	Ring descriptors
MAXDP	Maximal electrotopological positive variation	Topological indices
Psi_i_A	Intrinsic state pseudoconnectivity index - type S average	Topological indices
Yindex	Balaban Y index	Information indices
CIC4	Complementary Information Content index (neighborhood symmetry of 4-order)	Information indices
CIC5	Complementary Information Content index (neighborhood symmetry of 5-order)	Information indices
VR1_D/Dt	Randic-like eigenvector-based index from distance/detour matrix	2D matrix-based descriptors
SpDiam_B(m)	spectral diameter from Burden matrix weighted by mass	2D matrix-based descriptors
ATSC2m	Centred Broto-Moreau autocorrelation of lag 2 weighted by mass	2D autocorrelations
ATSC1p	Centred Broto-Moreau autocorrelation of lag 1 weighted by polarizability	2D autocorrelations
GATS6m	Geary autocorrelation of lag 6 weighted by mass	2D autocorrelations
GATS7s	Geary autocorrelation of lag 7 weighted by I-state	2D autocorrelations
P_VSA_LogP_1	P_VSA-like on LogP, bin 1	P_VSA-like descriptors
P_VSA_LogP_2	P_VSA-like on LogP, bin 2	P_VSA-like descriptors
P_VSA_LogP_8	P_VSA-like on LogP, bin 8	P_VSA-like descriptors
P_VSA_MR_3	P_VSA-like on Molar Refractivity, bin 3	P_VSA-like descriptors
T(N..Cl)	Sum of topological distances between N..Cl	2D Atom Pairs
T(O..F)	Sum of topological distances between O..F	2D Atom Pairs
T(O..Cl)	Sum of topological distances between O..Cl	2D Atom Pairs
T(F..Cl)	Sum of topological distances between F..Cl	2D Atom Pairs
F03[C-O]	Frequency of C - O at topological distance 3	2D Atom Pairs
F03[C-Cl]	Frequency of C - Cl at topological distance 3	2D Atom Pairs
MLOGP2	Squared Moriguchi octanol-water partition coeff. ($logP^2$)	Molecular properties

Table 4.10: QSPR model configuration and training hyperparameters

Model Component	Configuration / Parameters
Input features	52 descriptors
Hidden layers	Linear(52, 16) → Linear(16, 16)
Output layer	Linear(16, 1)
Activation function	ReLU
Total trainable parameters	1,137
Output	Predicted climate change category
Optimizer	Adam
Learning rate	5e-6
Batch size	20
Decreasing factor (LR scheduler)	0.9
Learning rate decay patience	10 epochs
Training epochs	800
Early stop patience	15 epochs

Table 4.11: GNN-M model configuration and training hyperparameters

Model Component	**Configuration / Parameters**
Input features	Molecular graphs
Input layer	Linear(32, 64) (2,112 params)
Edge network (NN)	Linear(13, 128) + Linear(128, 4096) (530,176 params)
Graph convolution	NNConv (parameters in NN)
Recurrent layer	GRU(64, 64) (24,768 params)
Fully connected head	Linear(64, 256) → Linear(256, 128) → Linear(128, 64) → Linear(64, 1) (Total: 57,857 params)
Output	Predicted climate change category
Total trainable parameters	**614,913**
Optimizer	Adam
Learning rate	0.0001
Batch size	20
Decreasing factor (LR scheduler)	0.9
Learning rate decay patience	10 epochs
Training epochs	500
Early stop patience	15 epochs
Activation functions	ReLU (NN), ELU (GNN layers)
Number of message passing steps	3
Pooling method	Add
Weight initialization	Normal(0, 0.1)

Table 4.12: Single-task GNN-E model configuration and training hyperparameters

Model Component	**Configuration / Parameters**
Input features	Molecular graphs and energy-mix features
Input layer	Linear(32, 64) (2,112 params)
Edge network (NN)	Linear(13, 128) + Linear(128, 4096) (530,176 params)
Graph convolution	NNConv (parameters in NN)
Recurrent layer	GRU(64, 64) (24,768 params)
Fully connected head	Linear(71, 256) → Linear(256, 128) → Linear(128, 64) → Linear(64, 1) (Total: 59,649 params)
Output	Predicted single impact category
Total trainable parameters	**616,705**
Optimizer	Adam
Batch size	20
Decreasing factor (LR scheduler)	0.9
Learning rate decay patience	10 epochs
Training epochs	500
Early stop patience	15 epochs
Activation functions	ReLU (NN), ELU (GNN layers)
Number of message passing steps	3
Pooling method	Add
Weight initialization	Normal(0, 0.1)

Table 4.13: Single-task GNN-C model configuration and training hyperparameters

Model Component	**Configuration / Parameters**
Input features	Molecular graphs and country categorical features
Input layer	Linear(32, 64) (2,112 params)
Edge network (NN)	Linear(13, 128) + Linear(128, 4096) (530,176 params)
Graph convolution	NNConv (parameters in NN)
Recurrent layer	GRU(64, 64) (24,768 params)
Fully connected head	Linear(155, 256) → Linear(256, 128) → Linear(128, 64) → Linear(64, 1) (Total: 81,153 params)
Output	Predicted single impact category
Total trainable parameters	**638,209**
Optimizer	Adam
Batch size	20
Decreasing factor (LR scheduler)	0.9
Learning rate decay patience	10 epochs
Training epochs	500
Early stop patience	15 epochs
Activation functions	ReLU (NN), ELU (GNN layers)
Number of message passing steps	3
Pooling method	Add
Weight initialization	Normal(0, 0.1)

Table 4.14: Multi-task GNN-C model configuration and training hyperparameters

Model Component	**Configuration / Parameters**
Input features	Molecular graphs, and country categorical features
Input layer	Linear(32, 64) (2,112 params)
Edge network (NN)	Linear(13, 128) + Linear(128, 4096) (530,176 params)
Graph convolution	NNConv (parameters in NN)
Recurrent layer	GRU(64, 64) (24,768 params)
Multi-task output head	15 MLPs with shared structure: Linear(155, 256) → Linear(256, 128) → Linear(128, 64) → Linear(64, 1) (Total: 1,217,295 params)
Output	Predicted all 15 impact categories
Total trainable parameters	**1,774,351**
Optimizer	Adam
Learning rate	0.001
Batch size	20
Decreasing factor (LR scheduler)	0.9
Learning rate decay patience	10 epochs
Training epochs	500
Early stop patience	15 epochs
Activation functions	ReLU (edge net), ELU (GNN + MLPs)
Number of message passing steps	3
Pooling method	Add
Weight initialization	Normal(0, 0.1)

Table 4.15: Multi-task GNN-E model configuration and training hyperparameters

Model Component	**Configuration / Parameters**
Input features	Molecular graphs and energy-mix features
Input layer	Linear(32, 64) (2,112 params)
Edge network (NN)	Linear(13, 128) + Linear(128, 4096) (530,176 params)
Graph convolution	NNConv (parameters in NN)
Recurrent layer	GRU(64, 64) (24,768 params)
Multi-task output head	15 MLPs with shared structure: Linear(71, 256) → Linear(256, 128) → Linear(128, 64) → Linear(64, 1) (Total: 894,735 params)
Output	Predicted all 15 impact categories
Total trainable parameters	**1,451,791**
Optimizer	Adam
Learning rate	0.001
Batch size	20
Decreasing factor (LR scheduler)	0.9
Learning rate decay patience	10 epochs
Training epochs	500
Early stop patience	15 epochs
Activation functions	ReLU (edge net), ELU (GNN + MLPs)
Number of message passing steps	3
Pooling method	Add
Weight initialization	Normal(0, 0.1)

Table 4.16: Category-specific learning rates for single-task GNN-C and GNN-E models

Impact Category	**GNN-C**	**GNN-E**
AC	1.00E–03	5.00E–04
CC	5.00E–05	5.00E–04
ECO	5.00E–04	1.00E–04
ER	5.00E–04	1.00E–04
EUf	1.00E–03	1.00E–03
EUm	5.00E–04	5.00E–04
EUt	5.00E–04	5.00E–04
HT	1.00E–03	1.00E–03
IR	1.00E–04	1.00E–03
LU	1.00E–04	1.00E–04
MR	1.00E–04	1.00E–03
OD	5.00E–04	5.00E–04
PMF	5.00E–04	5.00E–05
POF	5.00E–05	5.00E–05
WU	5.00E–04	5.00E–04

5

SELF-SUPERVISED GRAPH NEURAL NETWORKS FOR POLYMER PROPERTY PREDICTION

The estimation of polymer properties is of crucial importance in many domains such as energy, healthcare, and packaging. Recently, graph neural networks (GNNs) have shown promising results for the prediction of polymer properties based on supervised learning. However, the training of GNNs in a supervised learning task demands a huge amount of polymer property data that is time-consuming and computationally/experimentally expensive to obtain. Self-supervised learning offers great potential to reduce this data demand through pre-training the GNNs on polymer structure data only. These pre-trained GNNs can then be fine-tuned on the supervised property prediction task using a much smaller labeled dataset. We propose to leverage self-supervised learning techniques in GNNs for the prediction of polymer properties. We employ a recent polymer graph representation that includes essential features of polymers, such as monomer combinations, stochastic chain architecture, and monomer stoichiometry, and process the polymer graphs through a tailored GNN architecture. We investigate three self-supervised learning setups: (i) node- and edge-level pre-training, (ii) graph-level pre-training, and (iii) ensembled node-, edge- & graph-level pre-training. We additionally explore three different transfer strategies of fully connected layers with the GNN architecture. Our results indicate that the ensemble node-, edge- & graph-level self-supervised learning with all layers transferred depicts the best performance across dataset size. In scarce data scenarios, it decreases the root mean square errors by 28.39% and 19.09% for the prediction of electron affinity and ionization potential compared to supervised learning without the pre-training task.

This chapter has been published as "Self-supervised graph neural networks for polymer property prediction" by Gao, Q., Dukker, T., Schweidtmann, A. M., & Weber, J. M., Molecular Systems Design & Engineering, **9(11)**, 1130-1143 (2024)

5.1. INTRODUCTION

The rise of many novel technologies demands the development of innovative polymers, with applications such as polymer-based organic batteries [1], polymer photocatalysts that aid hydrogen generation [2], or more sustainable plastics [3]. However, navigating the vast chemical space to identify polymers with optimal properties poses a considerable challenge. Consequently, various computational methods have been proposed to assist preliminary screening tasks.

One widely recognized class of methods is quantitative structure-property relationships (QSPR) [4], which link molecules' physical or chemical properties to their structural information through the use of predefined molecular descriptors and linear or nonlinear regression techniques. Nonetheless, QSPR models carry intrinsic limitations. For instance, significant domain expertise is required to select appropriate molecular descriptors and potentially sub-optimal choices impact their prediction capacity. Additionally, the feature selection process needs to be repeated for each target property. These limitations are addressed by end-to-end learning methods in machine learning (ML), bypassing the need for manual selection of descriptors and enabling the models to discern relevant features autonomously. Here, the algorithms learn a molecular fingerprint directly from the molecular representation without the need for descriptor selection. Graph neural networks (GNNs) have recently emerged as a key end-to-end learning technique for molecular property prediction tasks, as they leverage the graphical representation of molecules [5]. Particularly, atoms are represented as nodes, and bonds are represented as edges. GNNs have shown to be very successful in a broad range of molecular prediction tasks, such as predicting combustion-related properties [6], predicting the growth of Escherichia coli for new antibiotic discovery [7], and protein function [8] or protein-protein interaction tasks [9].

Despite the success of GNNs in molecular machine learning, extending them to polymer property prediction remains challenging. The reasons for that are twofold. Firstly, polymer structures are difficult to model accurately. They exhibit an inherent hierarchical complexity and they are not one uniquely defined molecular structure but are composed of an ensemble of similar molecules [10]. Secondly, unlike in the domains of small molecules and ordered materials, where numerous large (>100K) and open-source databases exist [11, 12], there is an absence of accessible polymer property data [13–15]. The recent work by Aldeghi et al. [10], has represented polymers with selected higher-order structural information as a stochastic graph, making them applicable for GNNs. However, the second obstacle, the lack of sufficient polymer property data is still challenging. The measurements of polymer properties in the community rely highly on the instruments in the experiments. However, those instruments have few open interfaces and data models, hindering further developments of large property

databasets [16–18].

To address the scarcity of labeled data, several strategies have been proposed, including transfer learning [19, 20] and multi-task learning [21, 22]. For instance, Zhang et al. [19] pre-trained a transformer model on a dataset of 1 billion small molecules and subsequently fine-tuned the model with a small amount of labeled polymer property data (ranging from 300 to 3,000 samples). The resulting accuracies were comparable to those achieved by the TransPolymer [23] and polyBERT [24] models, without the expensive generation and pretraining on large-scale augmented polymer datasets. Furthermore, the objective of a multi-task training approach is to enhance the performance and generalization of a model by concurrently training it on multiple related tasks, thereby leveraging shared information and representations. Gurnani et al. [21] introduced multi-task GNNs for predicting 34 polymer properties. Their results indicated that, for datasets with fewer than 300 samples, the proposed models generally outperform the benchmark models.

Self-supervised learning (SSL) is another method for learning from limited data and currently helps solve data scarcity challenges in many fields like computer vision [25, 26] and natural language processing [27]. SSL first designs a predictive learning task within the unlabelled data itself such that the model learns universal features and semantics, which subsequently contribute to the downstream supervised task with less labeled training data. A common SSL technique, for example, is to create a pseudo label from the unlabelled data structure. Unlike other training strategies that incorporate pseudo-labels into the input to enhance performance [28], SSL pre-trains the model using pseudo-labels to extract initial semantic information from unlabeled data. This approach reduces the amount of labeled data needed for subsequent fine-tuning tasks. Taking computer vision as an example [29], one can rotate an image and task the model with predicting the rotation angle. Thereby the model learns certain features from images and their orientation in space; semantics that are useful for the downstream task of interest. In the field of graph-based molecular property prediction, such tasks are commonly designed according to the structure of the molecular graph. For example, Zhang et al. [30] leveraged node and edge masking as an SSL technique to pre-train GNNs, which improves the performance of predicting properties of organic semiconductors. Zang et al. [31] implemented a Hierarchical Molecular Graph Neural Network (HMGNN) with multi-level SSL techniques to achieve superior molecular property prediction results on both classification and regression tasks. With various successful applications in small molecular property prediction [30–32] and proteins [33, 34], SSL is a promising technique for prediction tasks in the sparse data domain of polymer informatics. Several studies [19, 23, 24, 33] have leveraged Large Language Models (LLMs) for polymer

property prediction tasks in an SSL manner. For example, polyBERT is pre-trained using masked language modeling (MLM) on approximately 80 million polymer strings, generated from about 13,800 original polymers. During fine-tuning, the pre-trained transformer weights are frozen and employed to provide polymer embeddings, which serve as inputs to a multi-layer perceptron for multitask property prediction. While these studies achieve similar performance, they are limited by using the PSMILES language, which only specifies the repeating unit and neglects important structural information such as chain architecture, stoichiometry, and monomer bonding, crucial for determining polymer properties.

We propose self-supervised GNNs for polymer property prediction. In particular, we utilize the state-of-the-art polymer graph representation and tailored GNN model from Aldeghi et al. [10] as foundation, and investigate three self-supervised learning setups: (i) pre-training with node- and edge-level pseudo tasks, (ii) pre-training with graph-level pseudo tasks, and (iii) ensembled node-, edge-, and graph-level pre-training. Additionally, we investigate various strategies for transferring the model architecture to augment the predictive accuracy, providing a novel perspective in the realm of polymer property prediction.

5.2. METHODS

This section introduces the methodologies employed throughout our work. First, the graph representation of polymers is introduced in Section 5.2.1 and the GNN architecture with tailored message-passing layers is illustrated in Section 5.2.2 (adapted from Aldeghi et al. [10]). Then, three different strategies of self-supervised learning are explained: node- and edge-level, graph-level task, and node- and edge- and graph-level task consecutively. Moreover, we outline our weight transferal strategies. Finally, we introduce the details of the polymer datasets and the training procedure.

5.2.1. POLYMER GRAPHS

We make use of a polymer representation that combines the graph representation of monomers with stochastic edges according to linking probabilities. We denote the graph representation of a polymer p with nodes as atoms $v \in N$ and edges as bonds $e_{vu} \in E$ connecting two nodes v and u. Specifically, there are two directed bonds between nodes v and u: bond e_{vu} and bond e_{uv}. Two directed edges allow for a wider variety of polymer structures. For example, as Figure 5.1 shows, they can capture the impact of terminal ends when required such as oligomer with termini, and different possibilities of two atoms being adjacent to one another such as graft copolymers. Moreover, a feature vector and a stochastic weight are assigned to each node and each directed edge. We denote the feature vector and the stochastic weight of node v with $\mathbf{f}^v \in F^N$ and $w_v \in W^N$, respectively. $\mathbf{f}^v$

contains specific information for atoms such as the atom type and w_v represents the stochiometric ratio of the corresponding monomer. Similarly, for the directed edge from node v to node u, a feature vector $\mathbf{f}^{e_{vu}} \in F^E$ and a stochastic weight $w_{vu} \in W^E$ are attached. $\mathbf{f}^{e_{vu}}$ contains specific information for bonds such as the bond type. $w_v \in (0,1]$ is an associated weight value reflecting the probability of that bond occurring within the respective polymer topologies. Notably, within the monomer, the edge weight is set to 1. While for the directed edges connecting different monomers, the weights vary as shown in Figure 5.1. Typical features are given in Table 5.1 for atoms and in Table 5.2 for bonds. The feature-enriched graph of a polymer p is then denoted by $G(p) = \{V, E, F^N, F^E, W^N, W^E\}$.

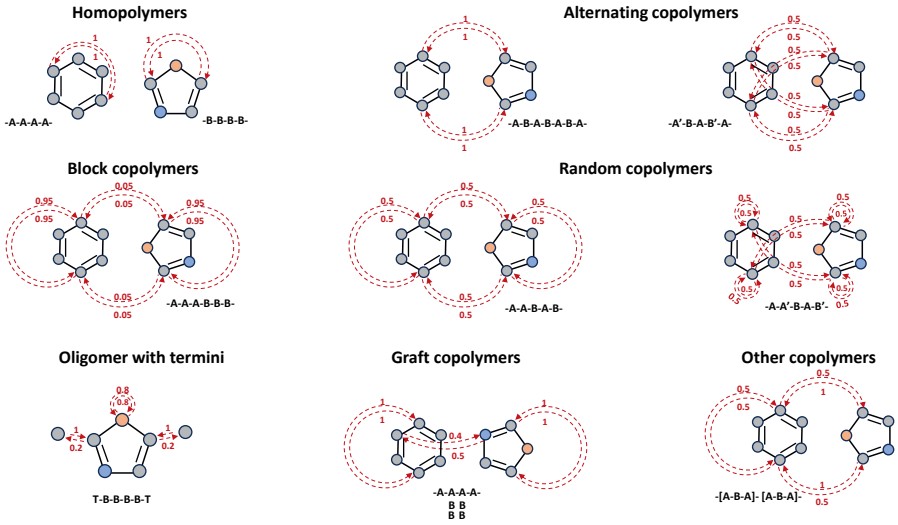


Figure 5.1: Overview of polymer topologies (adapted from Aldeghi et al. [10]). The red dashed lines represent the direct edges which are assigned to corresponding weights between zero and one. The weights of incoming edges indicate the possibility that the originating node (atom) is adjacent to the target node (atom). For instance, in the case of graft copolymers, there is a 40% chance that the atoms on monomer A will be neighboring atoms on monomer B (two of five monomers A are connected to monomers B), whereas the atoms on monomer B have a 50% chance of being adjacent to atoms on monomer A (two out of four monomers B are connected to the monomers A). The text underneath of each topology represents an example of the corresponding polymer sequence, where A and B are different monomers. The polymer sequence depicts a tail-to-head orientation, e.g., $-A-B-$ represents the tail of monomer A links to the head of monomer B, and $-A'-B-$ indicates the head of monomer A connects to the head of monomer B.

5.2.2. GRAPH NEURAL NETWORKS

GNNs can learn desired target predictions from a featured graph representation. Figure 5.2 shows an example of the GNN architecture that is used in this work

Table 5.1: Atom features for initial node feature vector [10]. Features are encoded as one-hot vectors except for atom mass.

Feature	Description
atom type	type of atom
chiral tag	type of chirality
degree	number of bonded neighbors in the graph
is aromatic	whether the atom is part of an aromatic system
hybridization	type of hybridization such as sp, sp2
charge	formal charge of the atom
# Hs	number of bonded hydrogen atoms
mass	atom mass

Table 5.2: Bond features for all initial edge feature vector [10]. Features are encoded as one-hot vectors.

Feature	Description
bond type	single, double, triple, or aromatic
conjugated	whether the bond is conjugated
is in a ring	whether the bond is part of a ring
stereo	none, any, E/Z, or cis/trans

for supervised learning-only training. GNNs consist of two phases: The message passing phase, which contains convolutional operations, and the readout phase, which forms the fingerprint and predicts the target task. In the following subsection, we will introduce these two phases separately.

In the *message passing* phase, graph convolution is utilized to extract the structural information from the polymer graph. Importantly, the weighted direct message passing neural network (wD-MPNN) is used as the graph convolutional layer. Figure 5.3 shows the four steps in wD-MPNN to extract structural information from polymers. In the first step, taking node v as an example, the edge feature $\mathbf{f}^{e_{vu}}$ from node v to node u is updated into $\mathbf{h}^0_{vu}$ as Equation 5.1 shows.

$$\mathbf{h}^0_{vu} = \tau(W_i(\mathbf{f}^v || \mathbf{f}^{e_{vu}})) \tag{5.1}$$

where $||$ denotes concatenation, W_i is a learnable weight matrix, and τ is an activation function. In the next step, namely edge-centered message passing, the hidden edge feature $\mathbf{h}^0_{vu}$ is further updated by L graph convolutional layers. Specifically, within $l \in L$, the new hidden edge feature $\mathbf{h}^{l=1}_{vu}$ is obtained by incoming edges to node v except the one from node u as Euqation 5.2.

$$\mathbf{h}^{l+1}_{vu} = \tau(\mathbf{h}^0_{vu} + W_h \sum_{k \in \{V(v)/u\}} w_{kv}\mathbf{h}^l_{kv}) \tag{5.2}$$

where W_i is a learnable weight matrix, and τ is an activation function. Furthermore, after L layers of edge-centered message passing, the updated node feature

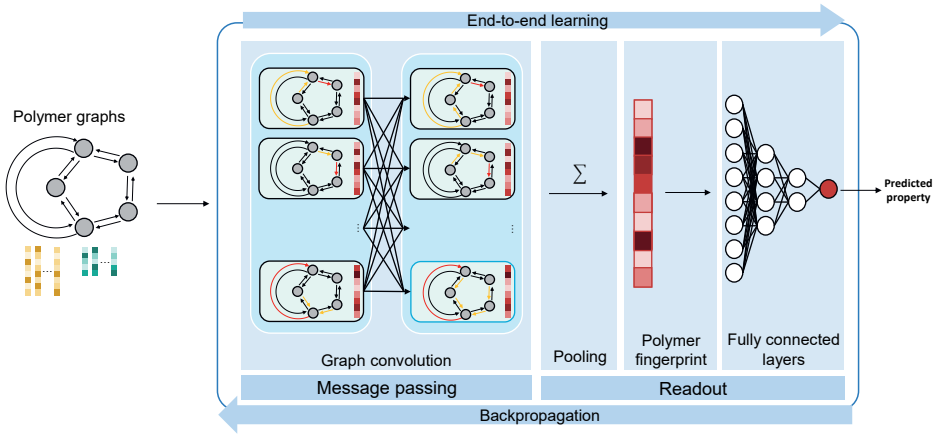


Figure 5.2: Overview of a GNN architecture for polymer property predictions.

$\mathbf{h}_v$ is acquired by taking the concatenation of the original node feature vector $\mathbf{f}^v$, with the weighted sum of the hidden edge feature vectors $\mathbf{h}_{kv}^L$ incoming to node v and passing this through a single fully connected layer. Equation 5.3 shows all steps where W_0 is a learnable weight matrix, and τ is an activation function. Notably, the edge-centered message passing as introduced by Aldeghi et al. [10] differentiates from node-centered message passing by previous works [35, 36].

$$\mathbf{h}_v = \tau(W_o(\mathbf{f}^v || \sum_{k \in V(v)} w_{kv} \mathbf{h}_{kv}^L)) \tag{5.3}$$

In the *readout* phase, the learned information from the previous steps in the hidden nodes is aggregated into a polymer fingerprint by taking the weighted average/sum as Equation 5.4 shows, where N is the number of nodes in the graph, w_v are the associated stochastic weights of each node.

$$\mathbf{h} = \frac{1}{N} \sum_{v=1}^{N} w_v \mathbf{h}_v \text{ or } = \sum_{v=1}^{N} w_v \mathbf{h}_v \tag{5.4}$$

Finally, the polymer fingerprint $\mathbf{h}$ is mapped into the property of interest. For GNNs, a multi-layer perception (MLP) is used to compute the property: $\hat{p} = \text{MLP}(\mathbf{h})$.

5.2.3. SELF-SUPERVISED LEARNING

Figure 5.4 illustrates the general framework of SSL: The pre-training phase where pseudo-tasks are designed for larger, yet unlabelled, datasets and the fine-tuning

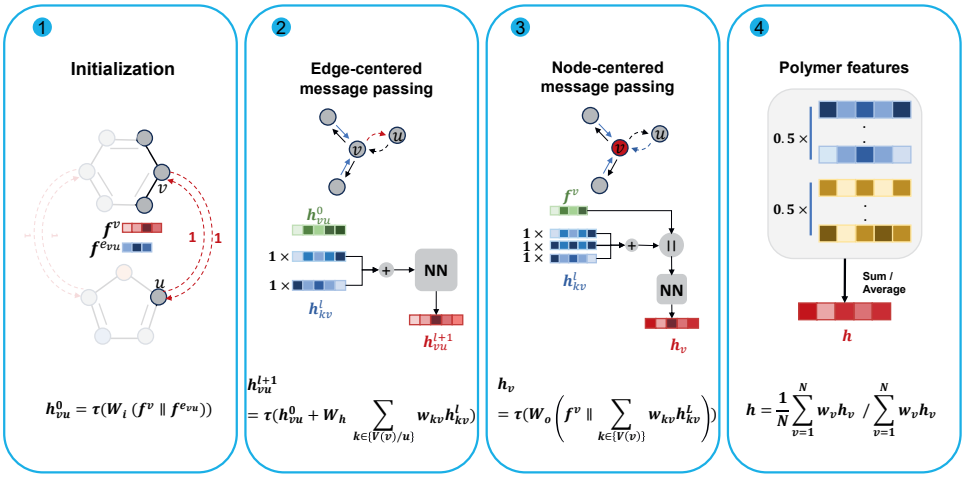


Figure 5.3: Four steps visualization of wD-MPNN [10]. First, the hidden edge features are initialized by concatenating originating atom features with edge features and then passing through a single fully connected layer. Second, hidden edge features are updated through L steps of edge-centered message passing. Within $l \in L$ steps, the neighboring hidden edge features are combined by the weighted sum, and then the results are added to the center hidden edge features $\mathbf{h}_{vu}^0$ and passed through a fully connected layer. Third, the center atom features f^v are updated through the concatenation of the weighted sum of incoming hidden edge features and the original atom features. The results are then passed through a single fully connected layer. Finally, a polymer fingerprint is obtained by taking the weighted average or sum of the updated atom feature vectors.

phase where the actual target labels are used for training purposes. During the pre-training phase, the encoder (message-passing layers in our case) is trained by polymer graphs with pseudo labels. These pseudo labels are obtained from the featured polymer graph topology. The aim is to design tasks for which no labeled data, i.e., physical properties, are needed to calculate these pseudo labels. Then, the pre-trained message-passing layers (including steps 1-4 in Figure 5.3) are further fine-tuned with or partially with an additional new readout training (e.g., new fully connected layers) by the downstream task with a small amount of labeled data in the second phase. Three SSL techniques are used in this work: (i) node- and edge-level, (ii) graph-level task, and (iii) node- and edge-level with consecutive graph-level tasks.

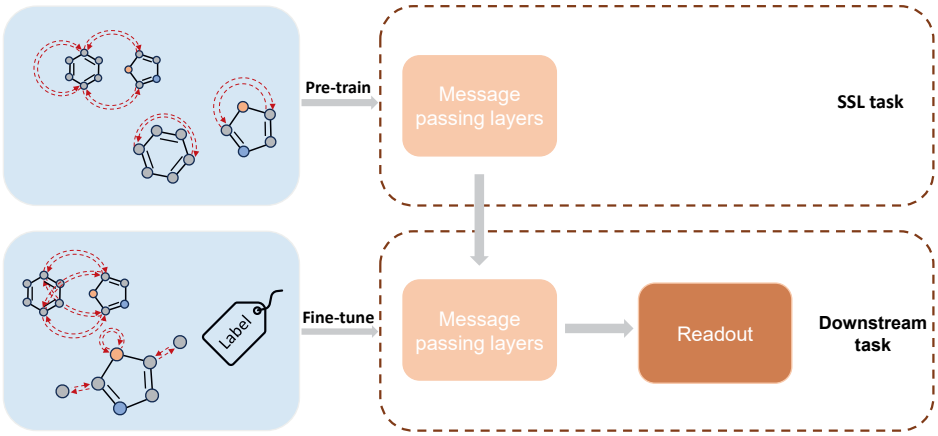


Figure 5.4: General framework of SSL [37]. The message passing layers are first trained with SSL tasks. Then the trained message passing layers are transferred with/partially with a new readout training (e.g., new fully connected layers) to train with target labels for downstream tasks.

NODE- AND EDGE-LEVEL TASK

We deploy node and edge masking at this level which refers to the practice of intentionally hiding a subset of node or edge information from the model. The model is trained to predict the missing information (pseudo label) from the masked graphs. This approach allows the model to learn the intrinsic structural information associated with surrounding nodes and edges, potentially grasping elementary chemical principles such as valence, and possibly more advanced concepts like functional group configurations. Figure 5.5 illustrates the workflow of node- and edge-level SSL and pseudo label prediction during training. Specifically, two node and two

edge feature vectors are randomly selected and masked by nullifying their feature vectors. Then, we leverage wD-MPNN (including steps 1-3 in Figure 5.3) to learn the structural information from the polymer graph, thereby obtaining learned node- and edge-level embeddings. Note that the prediction involves no pooling step as the pseudo labels are predicted from node- and edge-level information and not from graph-level information. Finally, taking the node- and edge-level embeddings as input, we use fully connected layers to predict the values of originally masked nodes and edges. We utilized the root mean square error (RMSE) to evaluate the performance. Finally, the pre-trained encoder together with the new fully connected layers is fine-tuned with target labels for the downstream task. The extent of layer transferal will be discussed in Section 5.2.3.

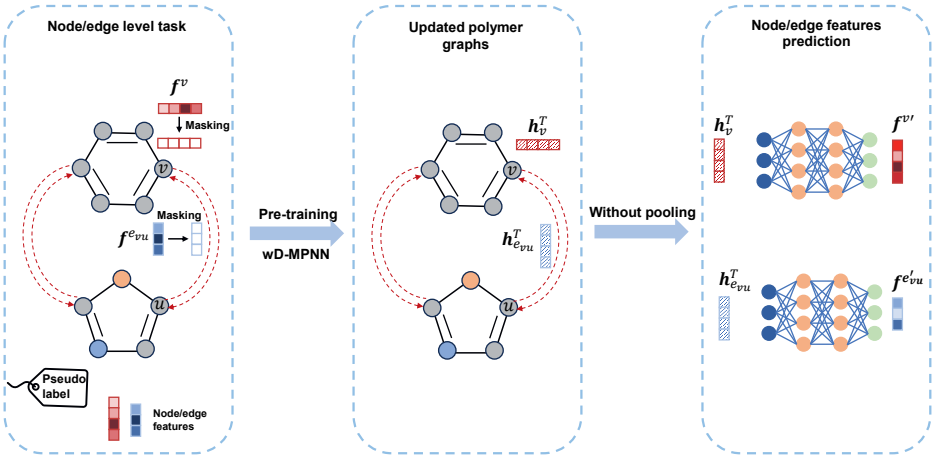


Figure 5.5: Overview of node/edge masking task. Randomly selected node and edge feature vectors are masked, then new embeddings are learned through the message passing phase, and finally, the originally masked vectors are predicted from the learned embeddings.

GRAPH-LEVEL TASK

The graph-level task aims at predicting a relevant pseudo label from graph-level information. As before with the node- and edge-level tasks the purpose is to pre-train the model with unlabelled data. The aim is that the graph-level pre-training can foster an understanding of overarching patterns, structural motifs, and inter-node relationships. Importantly, the aim is that through pre-training, the model can generalize better across various graph configurations and perform better in predicting target properties with less data. Figure 5.6 shows the overview of the graph-level SSL task. We propose to use an ensemble polymer molecular

weight that averages the molecular weight of the monomers (M_{mono}) with the corresponding stoichiometric ratio (w_{mono}) as Equation 5.5 shows.

$$M_{ensemble} = w_{mono1} M_{mono1} + w_{mono2} M_{mono2} \tag{5.5}$$

Specifically, we calculate the ensemble polymer weight for each data sample and then train the GNN with this new pseudo label as a predictive task. We utilize sum pooling layers to map the updated polymer graphs into a vector, i.e., the polymer fingerprint, which is then further mapped into the prediction of the pseudo graph label through fully connected layers. After the pre-training phase, as in the node- and edge-level pre-training, the pre-trained encoder together with the new fully connected layers is fine-tuned with the target labels for the downstream prediction task. The extent of layer transferal will be discussed in Section 5.2.3.

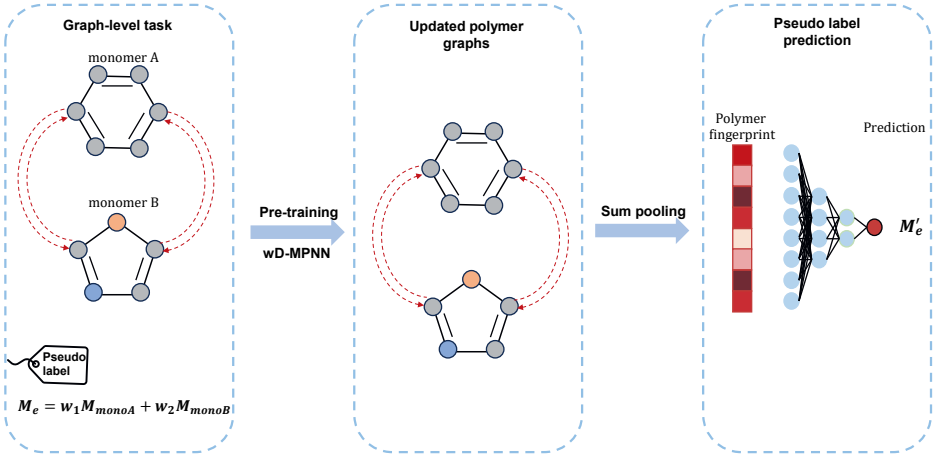


Figure 5.6: Overview of the graph-level task. Per co-polymer graph, the ensemble polymer molecular weight is calculated as a label, and then the message passing and readout phases take place to predict the pseudo label.

NODE-, EDGE- & GRAPH-LEVEL TASK

Our third strategy combines all the node-, edge-, and graph-level tasks consecutively. The encoder is first trained with the node- and edge-level task. Then, the encoder is transferred to perform the graph-level task with new fully connected layers. Ultimately, we transfer the encoder to predict the property of interest with new fully connected layers as the prediction head.

TRANSFER OF WEIGHTS

Finally, we explore the impacts of transferring the fully connected layers of the readout phase from the pre-training phase to the fine-tuning stage for the graph-level task and the combined approach of node-, edge-, and graph-level tasks. In particular, we examined three distinct transfer strategies: exclusive transfer of message-passing layers (strategy a), transfer of both message-passing and two fully connected layers (strategy b), and message-passing and all fully connected layers (strategy c).

5.2.4. POLYMER DATASET

We develop our method on the polymer dataset presented by Aldeghi et al. [10], encompassing 42,966 polymers, composed of nine A monomers and 862 B monomers. These polymers are categorized based on three distinct chain architectures: alternating, random, and block, and are further characterized by three stoichiometric ratios, namely 1:1, 1:3, and 3:1. Furthermore, the dataset contains electron affinity (EA) and ionization potential (IP) as molecular properties, i.e., target labels. Both properties were derived through Density Functional Theory (DFT) simulations.

5.2.5. TRAINING PROCEDURE

First, we trained two GNN models via a supervised learning paradigm, targeting both the EA and IP labels (in the following referred to as baseline training). Subsequently, we perform three SSL techniques: (i) node- and edge-level, (ii) graph-level task, (iii) node-, edge-, and graph-level tasks. The polymer dataset is randomly shuffled and 40% of the dataset is used for SSL pre-training tasks. We then transferred the message-passing layers and subsequently conducted exploratory analyses across varying dataset sizes for fine-tuning, spanning 0.08%, 0.16%, 0.24%, 0.32%, 0.4%, 0.8%, 1.6%, 2.4%, 3.2%, 4%, 8%, 16%, 24%, 32%, and up to 40%, allowing us to explore the performance of SSL under varying label availability scenarios The baseline models are only trained with the same fine-tuning datasets ranging from 0.08% to 40% in a supervised-only manner. Finally, the evaluation of the GNN models is conducted on the remaining 20% of the dataset. The Adam optimizer is used with a 10^{-3} learning rate and a batch size of 64 across 100 training epochs is used. The GNN architectures incorporate three edge-focused message-passing layers with a hidden dimension calibrated at 300. Additionally, three fully connected layers are used to map the fingerprints to the prediction task.

5.3. RESULTS AND DISCUSSION

In this section, we present the results from our three SSL techniques: (i) node- and edge-level, (ii) graph-level task, and (iii) a sequential application of node- and edge-level followed by the graph-level task. We also outline the results with different layer transferal strategies.

5.3.1. SSL RESULTS

Figure 5.7 illustrates the model performance metrics R^2 and $RMSE$ regarding electron affinity and ionization potential. The corresponding values and 95% confidence interval values are also detailed in the electronic supplementary information (ESI). In the small labeled data regimes (e.g., < 0.4%), node- and edge-level SSL, as well as node-, edge- & graph-level SSL, demonstrate similar performance across all evaluated models for both EA and IP according to the R^2 and $RMSE$. For instance, for node-, edge- & graph-level SSL, an increase of 32.21% and 23.21% in R^2 is observed for both EA and IP, respectively, compared with baseline models in the smallest fine-tuning dataset (0.08%). Similarly, for node- and edge-level SSL, R^2 is improved by 38.66% and 38.04%, respectively, followed by graph-level SSL in the smallest fine-tuning dataset. Therefore, in the small labeled data regimes, SSL significantly enhances the prediction performance.

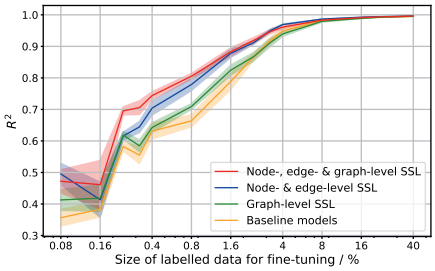
In the data regimes from 0.4% to 4%, node- and edge-level SSL, as well as node-, edge- & graph-level SSL, still boost the performance for both R^2 and $RMSE$ than baseline models. However, graph-level SSL exhibits less significant improvements and even negative contributions compared with baseline models. A potential reason is that the chosen pseudo label (ensemble molecular weight) has a discrepancy with target labels EA and IP.

As for larger labeled data regimes (e.g., > 4%), the differences between the models become smaller (potentially neglectable) regarding both performance metrics and property labels. A plausible explanation is that with larger fine-tuning dataset availability, the graph semantics can be learned directly through the target label in the supervised fine-tuning phase, which improves the final prediction together with the pre-training SSL phase.

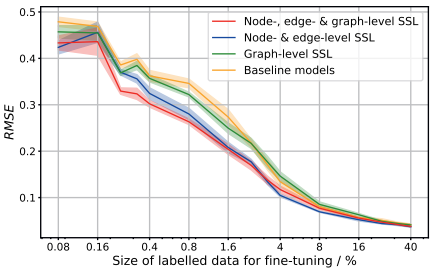
We further analyzed the error distribution of all four models for predicting EA and IP. First, it is evident that all error distributions follow a long-tail pattern, indicating that while the majority of predictions are accurate (low RMSE), there are a few significant outliers. In the smallest labeled data regime (0.08%), both for EA and IP, graph-level pre-training contributes minimally to performance improvement, as indicated by error distributions close to those of the baseline models. In contrast, the node-, edge-level, and combined node-, edge-, and graph-level SSL models demonstrate superior performance, with error distributions peaking over 500 with slightly lower RMSE and being narrower for both EA and IP.

Notably, for EA, all models struggle to predict polymers containing Pyromellitic dianhydride (PMDA) as Figure5.7a shows. This difficulty likely arises from PMDA's strong electron-withdrawing properties due to the presence of anhydride groups, complicating EA predictions. Additionally, only 63 out of 42,966 samples contain PMDA, and this data scarcity contributes to poor generalization. Conversely, for IP, there is no common failure pattern in the predictions for specific polymers. In the larger labeled data regime (24%), significant improvements are observed for both EA and IP, with RMSE distributions becoming narrower. For both properties, SSL models continue to outperform baseline models, demonstrated by higher peaks with slightly lower RMSE and narrower error distributions.

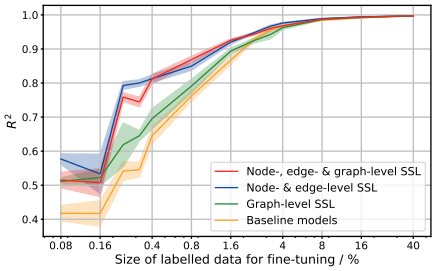
Overall, node- and edge-level SSL, as well as node-, edge- & graph-level SSL, exhibit consistently better performance than the baseline supervised-learning results even in the biggest labeled data regime. For example, the R^2 still increases by 0.1% and 0.1% for both EA and IP for node-, edge- & graph-level SSL.



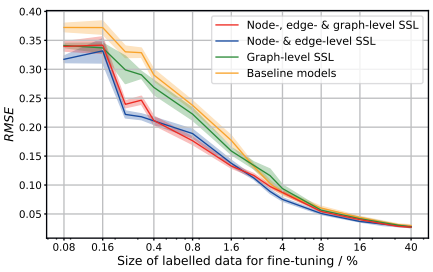
(a) R^2 results for EA



(b) $RMSE$ results for EA



(c) R^2 results for IP



(d) $RMSE$ results for IP

Figure 5.7: R^2 and $RMSE$ results for EA and IP for supervised learning only, node- and edge-level, graph-level task, and node-, edge- & graph-level, respectively. Each line is the mean from 10 individual runs and the shaded area is the 95% confidence interval.

5.3.2. TRANSFERRING FULLY CONNECTED LAYERS

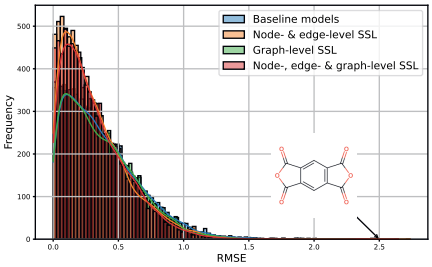
In this section, we present and discuss the impact of the layer transfer strategy within SSL for both graph-level SSL and node-, edge- & graph-level, respectively.

Table 5.3: R^2 of predicting the EA and IP labels using different sizes of datasets for the fine-tuning part and the corresponding standard deviation can be found in ESI. Specifically, we use baseline for representing baseline models, NE for representing node- & edge-level SSL, G for representing graph-level SSL, and NEG for representing node-, edge-, & graph-level SSL.

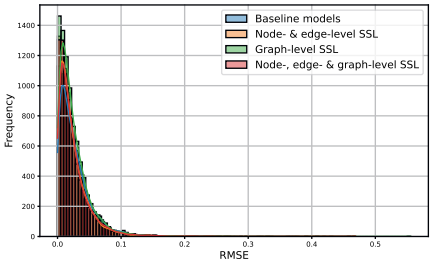
Dataset Size	R^2 of EA				R^2 of IP			
	Baseline	NE	G	NEG	Baseline	NE	G	NEG
0.08%	0.357	0.495	0.413	0.472	0.418	0.577	0.511	0.515
0.16%	0.383	0.413	0.418	0.461	0.417	0.533	0.522	0.508
0.24%	0.583	0.616	0.619	0.695	0.541	0.793	0.619	0.759
0.32%	0.555	0.644	0.585	0.706	0.546	0.800	0.645	0.744
0.4%	0.631	0.704	0.643	0.744	0.646	0.813	0.695	0.812
0.8%	0.664	0.779	0.709	0.805	0.762	0.850	0.791	0.869
1.6%	0.789	0.878	0.824	0.883	0.868	0.920	0.894	0.926
2.4%	0.867	0.912	0.867	0.919	0.927	0.947	0.924	0.943
3.2%	0.918	0.949	0.912	0.946	0.955	0.967	0.942	0.960
4%	0.949	0.969	0.939	0.961	0.968	0.976	0.963	0.968
8%	0.982	0.987	0.979	0.983	0.985	0.989	0.987	0.988
16%	0.991	0.992	0.989	0.991	0.992	0.994	0.993	0.993
24%	0.993	0.995	0.993	0.994	0.994	0.995	0.995	0.995
32%	0.994	0.995	0.995	0.995	0.994	0.996	0.995	0.996
40%	0.995	0.996	0.995	0.996	0.996	0.997	0.996	0.997

Table 5.4: *RMSE* of predicting the EA and IP labels using different sizes of datasets for the fine-tuning part and the corresponding standard deviation can be found in ESI. Specifically, we use baseline for representing baseline models, NE for representing node- & edge-level SSL, G for representing graph-level SSL, and NEG for representing node-, edge-, & graph-level SSL.

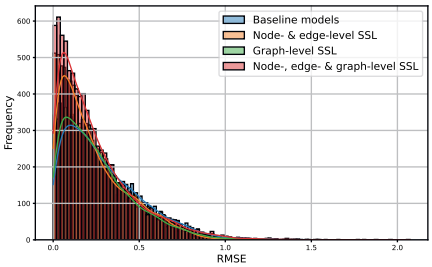
Dataset Size	*RMSE* of EA				*RMSE* of IP			
	Baseline	NE	G	NEG	Baseline	NE	G	NEG
0.08%	0.479	0.424	0.457	0.433	0.372	0.317	0.341	0.340
0.16%	0.469	0.456	0.455	0.436	0.372	0.332	0.337	0.342
0.24%	0.386	0.370	0.369	0.329	0.330	0.222	0.299	0.239
0.32%	0.398	0.356	0.385	0.323	0.329	0.218	0.290	0.247
0.4%	0.362	0.324	0.357	0.302	0.290	0.211	0.269	0.211
0.8%	0.346	0.280	0.322	0.263	0.238	0.189	0.223	0.177
1.6%	0.273	0.208	0.250	0.204	0.177	0.138	0.159	0.133
2.4%	0.217	0.177	0.217	0.169	0.132	0.112	0.134	0.116
3.2%	0.170	0.134	0.177	0.138	0.103	0.089	0.116	0.098
4%	0.134	0.105	0.146	0.117	0.086	0.075	0.094	0.087
8%	0.080	0.069	0.085	0.077	0.060	0.051	0.057	0.054
16%	0.057	0.052	0.063	0.056	0.044	0.037	0.042	0.040
24%	0.051	0.044	0.049	0.047	0.036	0.033	0.035	0.033
32%	0.045	0.042	0.044	0.042	0.031	0.029	0.031	0.029
40%	0.042	0.037	0.041	0.038	0.030	0.028	0.027	0.027



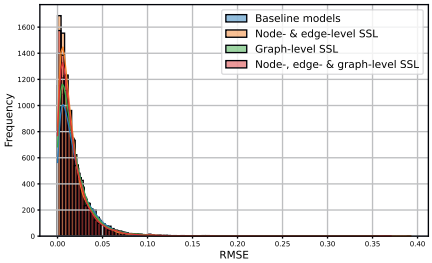
(a) 0.08% fine-tuning dataset for EA



(b) 24% fine-tuning dataset for EA



(c) 0.08% fine-tuning dataset for IP



(d) 24% fine-tuning dataset for IP

Figure 5.8: Error distribution for EA and IP with fine-tuning dataset 0.08% and 24%, respectively.

Figure 5.9 shows the *RMSE* results with our three different transfer strategies for both EA and IP. Here we only show the *RMSE* results as the R^2 renounces in the same trend.

In the labeled data regimes <4%, we observe a pronounced difference when transferring all layers (strategy c) for the prediction of the label EA in contrast to the baseline model and the two other strategies where fewer layers are transferred. This holds both for graph-level SSL and node-, edge- & graph-level SSL. For example, the *RMSE* significantly decreases by 15.10% and 20.79% compared to strategy a, where only message-passing layers are transferred, for both graph-level SSL and node-, edge- & graph-level SSL, respectively. Strategy a and b also outperform the baseline models in all low data availability scenarios, yet less significantly than strategy c. On the other hand, when predicting the IP, the three strategies for graph-level SSL have a similar performance in terms of *RMSE* and show a less significant improvement over the baseline model. For node-, edge-, and graph-level SSL, all three strategies still exhibit a similar performance but improve the prediction significantly compared to the baseline model. For example, the *RMSE* of strategy c decreases by 19.09% compared to the baseline model in the smallest data scenario (0.08%). This indicates that during the node- & edge-level SSL pre-training, the learned semantics are additionally embedded in the message passing layers, which significantly contributes to predicting the

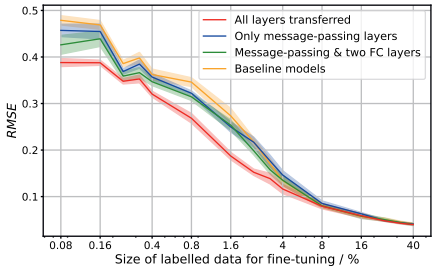
target labels in the downstream tasks.

In the labeled data regime (e.g., >4%), the impact of transferring fully connected layers becomes less significant: the importance of SSL decreases as shown by the performance curves approaching each other. For node-, edge- & graph-level SSL, strategy c is still consistently better than the baseline models for both EA and IP. However, for IP, we do observe a slightly negative contribution (4%) from strategy c compared to the baseline model for graph-level SSL. A plausible explanation is that graph-level SSL indeed has a negative contribution in the larger labeled data regime as we observed before. The chosen pseudo label from the pre-training phase is not sufficiently informative for the target label IP, and the learned semantics stored in the fully connected layer introduce extra bias during the pre-training and thereby hinder further learning in fine-tuning.

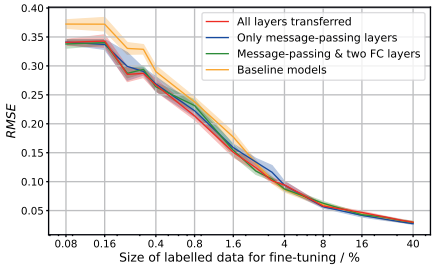
To sum it up, the extent of transferred weights impacts the overall performance of SSL in the two case studies investigated in our work. In small fine-tuning dataset scenarios (e.g., < 0.4%), all transferal strategies perform better than the baseline models in both prediction tasks and both SSL scenarios. For both EA and IP, strategy c where most layers were transferred demonstrates the best performance, which decreases the *RMSE* by 28.39% and 19.09% respectively, compared to the baseline models for node-, edge- & graph-level SSL. In larger fine-tuning dataset scenarios, the differences become less pronounced as the impact of SSL decreases. Here, scenario c, where most layers are transferred, also showed to perform slightly worse than the baseline model for IP. Thus, we showed that next to the design of the molecular-topology based pseudo-tasks, additional design decisions on the extent of layer transfer are important for designing best performing SSL pipelines.

5.4. CONCLUSIONS

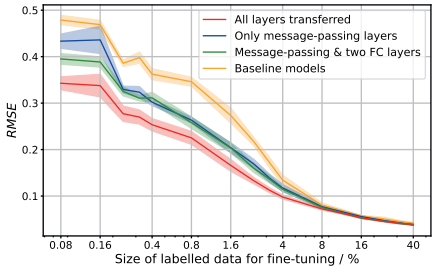
We propose a first SSL framework using GNNs for polymer property prediction to address the challenge of improved learning from small polymer datasets. We employ a state-of-the-art polymer graph representation and a wD-MPNN architecture to process polymer graphs. Our contribution investigates three different SSL techniques: (i) node- and edge-level SSL, (ii) graph-level SSL (iii) node-, edge- & graph-level SSL. Furthermore, we explore the impacts of different weight transfer strategies for the fully connected layers. The results show that SSL indeed boosts R^2 32.21% and 23.21% for EA and IP respectively, especially when labeled data is extremely limited (0.08%) for the fine-tuning phase (our two smallest data size scenarios). Additionally, SSL can still contribute to augmenting the R^2 by 0.1% and 0.1% for EA and IP, even in the largest labeled data regime. Moreover, we highlighted the impact of layer transferal during SSL, where transferring all fully connected layers was advantageous in small data scenarios. Overall, the ensemble



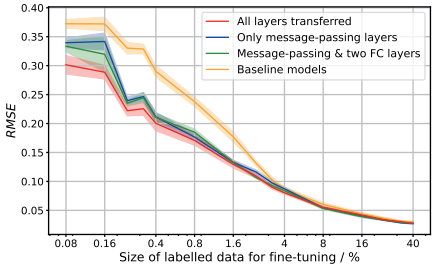
(a) Graph-level SSL of *RMSE* results for EA



(b) Graph-level SSL of *RMSE* results for IP



(c) Node-, edge- & graph-level SSL of *RMSE* results for EA



(d) Node-, edge- & graph-level SSL of *RMSE* results for IP

Figure 5.9: Graph-level and Node-, edge- & graph-level SSL of *RMSE* results for both EA and IP. The results are from supervised-only, transferring only message-passing layers (strategy a) indicated in green in all four subplots, transferring message-passing & two fully connected layers (strategy b) indicated in blue in all four subplots, and transferring all layers (strategy c) indicated in red in all four subplots. Each line is the mean from 10 individual runs and the shaded area is the 95% confidence interval.

Table 5.5: *RMSE* of predicting the EA and IP labels using different sizes of datasets for transferring different fully connected layers in graph-level SSL and the corresponding standard deviation can be found in ESI. Specifically, we use baseline for representing baseline models, G for representing graph-level SSL without transferring fully connected layers, G2 for representing graph-level SSL without transferring two fully connected layers, and G3 for representing graph-level SSL without transferring all fully connected layers.

Dataset Size	*RMSE* of EA				*RMSE* of IP			
	Baseline	G	G2	G3	Baseline	G	G2	G3
0.08%	0.479	0.457	0.426	0.388	0.372	0.341	0.338	0.341
0.16%	0.469	0.455	0.439	0.388	0.372	0.337	0.340	0.343
0.24%	0.386	0.369	0.359	0.348	0.330	0.299	0.287	0.285
0.32%	0.398	0.385	0.366	0.353	0.329	0.290	0.294	0.287
0.4%	0.362	0.357	0.347	0.320	0.290	0.269	0.264	0.267
0.8%	0.346	0.322	0.314	0.268	0.238	0.223	0.231	0.214
1.6%	0.273	0.250	0.253	0.187	0.177	0.159	0.154	0.151
2.4%	0.217	0.217	0.198	0.152	0.132	0.134	0.118	0.124
3.2%	0.170	0.177	0.158	0.139	0.103	0.116	0.104	0.102
4%	0.134	0.146	0.135	0.116	0.086	0.094	0.087	0.094
8%	0.080	0.085	0.080	0.079	0.060	0.057	0.063	0.058
16%	0.057	0.063	0.058	0.059	0.044	0.042	0.043	0.047
24%	0.051	0.049	0.051	0.048	0.036	0.035	0.039	0.039
32%	0.045	0.049	0.044	0.043	0.031	0.031	0.033	0.033
40%	0.042	0.041	0.040	0.040	0.030	0.027	0.030	0.030

Table 5.6: *RMSE* of predicting the EA and IP labels using different sizes of datasets for transferring different fully connected layers in node-, edge- & graph-level SSL and the corresponding standard deviation can be found in ESI. Specifically, we use baseline for representing baseline models, NEG for representing node-, edge- & graph-level SSL without transferring fully connected layers, NEG2 for representing node-, edge- & graph-level SSL without transferring two fully connected layers, and NEG3 for representing node-, edge- & graph-level SSL without transferring all fully connected layers.

Dataset Size	*RMSE* of EA				*RMSE* of IP			
	Baseline	NEG	NEG2	NEG3	Baseline	NEG	NEG2	NEG3
0.08%	0.479	0.433	0.395	0.343	0.372	0.340	0.333	0.301
0.16%	0.469	0.436	0.389	0.338	0.372	0.342	0.320	0.289
0.24%	0.386	0.329	0.325	0.277	0.330	0.239	0.235	0.222
0.32%	0.398	0.323	0.311	0.270	0.329	0.247	0.245	0.226
0.4%	0.362	0.302	0.311	0.253	0.290	0.211	0.211	0.200
0.8%	0.346	0.263	0.258	0.225	0.238	0.177	0.185	0.171
1.6%	0.273	0.204	0.204	0.166	0.177	0.133	0.133	0.129
2.4%	0.217	0.169	0.160	0.133	0.132	0.108	0.107	0.110
3.2%	0.170	0.138	0.136	0.112	0.103	0.093	0.093	0.090
4%	0.134	0.117	0.115	0.098	0.086	0.087	0.084	0.080
8%	0.080	0.077	0.076	0.072	0.060	0.054	0.053	0.055
16%	0.057	0.056	0.052	0.054	0.044	0.039	0.039	0.042
24%	0.051	0.047	0.046	0.045	0.036	0.033	0.033	0.034
32%	0.045	0.042	0.041	0.041	0.031	0.029	0.029	0.030
40%	0.042	0.038	0.038	0.039	0.030	0.028	0.028	0.028

node-, edge- & graph-level SSL with all transferred layers depicts the best performance. It decreases the *RMSE* by 28.39% and 19.09% for EA and IP, respectively. Our results contribute to well-reasoned and improved SSL pipelines in the field of polymer property prediction. Our current study introduces a single graph pseudo label and one node/edge masking strategy. Future research could explore the extension to multiple graph pseudo labels and diverse node/edge masking strategies. Additionally, subsequent work may consider integrating various datasets to evaluate the robustness of the proposed SSL framework. Moreover, a promising future investigation is the integration of SSL with Quantitative Structure-Activity Relationship (QSAR) or QSPR modeling for predicting polymer properties. For instance, one approach could involve masking a specific property within the descriptors and predicting the masked property during the pre-training phase. This pre-trained model could then be fine-tuned using the target property dataset.

REFERENCES

[1] S. Muench, A. Wild, C. Friebe, B. Häupler, T. Janoschka, and U. S. Schubert, *Polymer-based organic batteries,* Chemical Reviews **116**, 9438 (2016).

[2] Y. Wang, A. Vogel, M. Sachs, R. S. Sprick, L. Wilbraham, S. J. A. Moniz, R. Godin, M. A. Zwijnenburg, J. R. Durrant, A. I. Cooper, and J. Tang, *Current understanding and challenges of solar-driven hydrogen generation using polymeric photocatalysts,* Nature Energy **4**, 746 (2019).

[3] Y. Zhu, C. Romain, and C. K. Williams, *Sustainable polymers from renewable resources,* Nature **540**, 354 (2016).

[4] L. P. Hammett, *Some relations between reaction rates and equilibrium constants.* Chemical Reviews **17**, 125 (1935).

[5] E. Heid, K. P. Greenman, Y. Chung, S.-C. Li, D. E. Graff, F. H. Vermeire, H. Wu, W. H. Green, and C. J. McGill, *Chemprop: A machine learning package for chemical property prediction,* Journal of Chemical Information and Modeling (2023).

[6] A. M. Schweidtmann, J. G. Rittig, A. König, M. Grohe, A. Mitsos, and M. Dahmen, *Graph neural networks for prediction of fuel ignition quality,* Energy & Fuels **34**, 11395 (2020).

[7] J. M. Stokes, K. Yang, K. Swanson, W. Jin, A. Cubillos-Ruiz, N. M. Donghia, C. R. MacNair, S. French, L. A. Carfrae, Z. Bloom-Ackermann, V. M. Tran, A. Chiappino-Pepe, A. H. Badran, I. W. Andrews, E. J. Chory, G. M. Church, E. D. Brown, T. S. Jaakkola, R. Barzilay, and J. J. Collins, *A deep learning approach to antibiotic discovery,* Cell **180**, 688 (2020).

[8] R. You, S. Yao, H. Mamitsuka, and S. Zhu, *DeepGraphGO: graph neural network for large-scale, multispecies protein function prediction,* Bioinformatics **37**, i262 (2021).

[9] H. Zhou, W. Wang, J. Jin, Z. Zheng, and B. Zhou, *Graph neural network for protein–protein interaction prediction: A comparative study,* Molecules **27**, 6135 (2022).

[10] M. Aldeghi and C. W. Coley, *A graph representation of molecular ensembles for polymer property prediction,* Chemical Science **13**, 10486 (2022).

[11] L. Ruddigkeit, R. van Deursen, L. C. Blum, and J.-L. Reymond, *Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17,* Journal of Chemical Information and Modeling **52**, 2864 (2012).

[12] J. J. Irwin and B. K. Shoichet, *Zinc - a free database of commercially available compounds for virtual screening,* Journal of Chemical Information and Modeling **45**, 177 (2004).

[13] C. Kim, A. Chandrasekaran, T. D. Huan, D. Das, and R. Ramprasad, *Polymer genome: A data-powered polymer informatics platform for property predictions,* The Journal of Physical Chemistry C **122**, 17575 (2018).

[14] S. Otsuka, I. Kuwajima, J. Hosoya, Y. Xu, and M. Yamazaki, *Polyinfo: Polymer database for polymeric materials design,* in *2011 International Conference on Emerging Intelligent Data and Web Technologies* (IEEE, 2011).

[15] Y. Amamoto, *Data-driven approaches for structure-property relationships in polymer science for prediction and understanding,* Polymer Journal **54**, 957 (2022).

[16] T. B. Martin and D. J. Audus, *Emerging trends in machine learning: A polymer perspective,* ACS Polymers Au **3**, 239 (2023).

[17] R. A. Patel, C. H. Borca, and M. A. Webb, *Featurization strategies for polymer sequence or composition design by machine learning,* Molecular Systems Design & Engineering **7**, 661 (2022).

[18] Y. Zhao, R. J. Mulder, S. Houshyar, and T. C. Le, *A review on the application of molecular descriptors and machine learning in polymer design,* Polymer Chemistry **14**, 3325 (2023).

[19] P. Zhang, L. Kearney, D. Bhowmik, Z. Fox, A. K. Naskar, and J. Gounley, *Transferring a molecular foundation model for polymer property predictions,* (2023), 10.48550/ARXIV.2310.16958, arXiv:2310.16958 [cs.LG] .

[20] H. Yamada, C. Liu, S. Wu, Y. Koyama, S. Ju, J. Shiomi, J. Morikawa, and R. Yoshida, *Predicting materials properties with little data using shotgun transfer learning,* ACS Central Science **5**, 1717 (2019).

[21] R. Gurnani, C. Kuenneth, A. Toland, and R. Ramprasad, *Polymer informatics at scale with multitask graph neural networks,* Chemistry of Materials **35**, 1560 (2023).

[22] O. Queen, G. A. McCarver, S. Thatigotla, B. P. Abolins, C. L. Brown, V. Maroulas, and K. D. Vogiatzis, *Polymer graph neural networks for multitask property learning,* npj Computational Materials **9** (2023), 10.1038/s41524-023-01034-3.

[23] C. Xu, Y. Wang, and A. Barati Farimani, *Transpolymer: a transformer-based language model for polymer property predictions,* npj Computational Materials **9** (2023), 10.1038/s41524-023-01016-5.

[24] C. Kuenneth and R. Ramprasad, *polybert: a chemical language model to enable fully machine-driven ultrafast polymer informatics,* Nature Communications **14** (2023), 10.1038/s41467-023-39868-6.

[25] I. Misra and L. van der Maaten, *Self-supervised learning of pretext-invariant representations,* in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2020).

[26] S. Azizi, B. Mustafa, F. Ryan, Z. Beaver, J. Freyberg, J. Deaton, A. Loh, A. Karthikesalingam, S. Kornblith, T. Chen, V. Natarajan, and M. Norouzi, *Big self-supervised models advance medical image classification,* in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (IEEE, 2021).

[27] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, *Albert: A lite bert for self-supervised learning of language representations,* (2019), 10.48550/ARXIV.1909.11942, arXiv:1909.11942 [cs.CL] .

[28] C. Brozos, J. G. Rittig, S. Bhattacharya, E. Akanny, C. Kohlmann, and A. Mitsos, *Predicting the temperature dependence of surfactant cmcs using graph neural networks,* Journal of Chemical Theory and Computation **20**, 5695 (2024).

[29] S. Gidaris, P. Singh, and N. Komodakis, *Unsupervised representation learning by predicting image rotations,* (2018), 10.48550/ARXIV.1803.07728, arXiv:1803.07728 [cs.CV] .

[30] Z. Zhang, Q. Liu, H. Wang, C. Lu, and C.-K. Lee, *Motif-based graph self-supervised learning for molecular property prediction,* (2021), 10.48550/ARXIV.2110.00987, arXiv:2110.00987 [q-bio.QM] .

[31] X. Zang, X. Zhao, and B. Tang, *Hierarchical molecular graph self-supervised learning for property prediction,* Communications Chemistry **6** (2023), 10.1038/s42004-023-00825-5.

[32] S. Chithrananda, G. Grand, and B. Ramsundar, *Chemberta: Large-scale self-supervised pretraining for molecular property prediction,* (2020), 10.48550/ARXIV.2010.09885, arXiv:2010.09885 [cs.LG] .

[33] A. X. Lu, H. Zhang, M. Ghassemi, and A. Moses, *Self-supervised contrastive learning of protein representations by mutual information maximization,* (2020), 10.1101/2020.09.04.283929.

[34] R. Krishnan, P. Rajpurkar, and E. J. Topol, *Self-supervised learning in medicine and healthcare,* Nature Biomedical Engineering **6**, 1346 (2022).

[35] M. Simonovsky and N. Komodakis, *Dynamic edge-conditioned filters in convolutional neural networks on graphs,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[36] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, *Neural message passing for quantum chemistry,* (2017), 10.48550/ARXIV.1704.01212, arXiv:1704.01212 [cs.LG] .

[37] Y. Xie, Z. Xu, J. Zhang, Z. Wang, and S. Ji, *Self-supervised learning of graph neural networks: A unified review,* (2021), 10.48550/ARXIV.2102.10757, arXiv:2102.10757 [cs.LG] .

6

BAYESIAN UNCERTAINTY QUANTIFICATION OF GRAPH NEURAL NETWORKS USING STOCHASTIC GRADIENT HAMILTONIAN MONTE CARLO

Graph neural networks (GNNs) have proven state-of-the-art performance in molecular property prediction tasks. However, a significant challenge with GNNs is the reliability of their predictions, particularly in critical domains where quantifying model confidence is essential. Therefore, assessing uncertainty in GNN predictions is crucial to improving their robustness. Existing uncertainty quantification methods, such as Deep ensembles and Monte Carlo Dropout, have been applied to GNNs with some success, but these methods are limited to approximating the full posterior distribution. In this work, we propose a novel approach for scalable uncertainty quantification in molecular property prediction using Stochastic Gradient Hamiltonian Monte Carlo (SGHMC). Additionally, we utilize a cyclical learning rate to facilitate sampling from multiple posterior modes, which improves posterior exploration within a single training round. Moreover, we compare the proposed methods with Monte Carlo Dropout and Deep ensembles, focusing on error analysis, calibration, and sharpness, considering both epistemic and aleatoric uncertainties. Our experimental results demonstrate that the proposed parallel-SGHMC approach significantly outperforms Monte Carlo Dropout and Deep ensembles in terms of calibration and sharpness. Specifically, parallel-SGHMC reduces the sum of squared errors by 99.4% and 75%, respectively, when compared to Monte Carlo Dropout and Deep Ensembles. These findings suggest that parallel-SGHMC is a promising method for uncertainty quantification in GNN-based molecular property prediction.

This chapter has been published as "Bayesian uncertainty quantification of graph neural networks using stochastic gradient Hamiltonian Monte Carlo" by Gao, Q., Miedema, D. C., Zhao, Y., Weber, J. M., Tao, Q., & Schweidtmann, A. M., Systems and Control Transactions, 1360-1364 (2024)

6.1. INTRODUCTION

Graph neural networks (GNNs) have demonstrated promising potential in accelerating the computational estimation of molecular properties [1–3]. However, their reliability remains a significant challenge, e.g., in safety-critical applications where understanding model confidence is crucial. Accurately assessing the uncertainty of GNN predictions is essential for enhancing their robustness, particularly when predicting complex molecular behaviors.

Uncertainty in machine learning can generally be divided into two types: aleatoric and epistemic uncertainty [4]. Aleatoric uncertainty arises from inherent noise in the data generation process, such as experimental measurement errors or environmental variations, and is considered irreducible. In contrast, epistemic uncertainty stems from the lack of knowledge of a model, often due to limited data or an incomplete understanding of the process, and can be reduced through additional data collection or improved model design. For instance, in modeling complex chemical reactions, using a more advanced model architecture—such as incorporating reaction kinetics and thermodynamic principles—can reduce epistemic uncertainty by capturing the process dynamics more accurately. Estimating both types of uncertainty is critical in domains like molecular property prediction, where errors can have substantial consequences.

Two popular methods for joint uncertainty quantification of machine learning models are Deep ensembles and Monte Carlo Dropout (MC-dropout). MC-dropout [5] applies dropout during both training and inference, generating multiple stochastic forward passes to approximate the posterior distribution of the model. Deep ensembles [6] involve training multiple independent models and averaging their predictions, with the variance among them reflecting the model uncertainty. Although both methods have been widely adopted for scalable uncertainty estimation in molecular property prediction [7], they have some limitations. Deep ensembles primarily focus on the maximum a posteriori (MAP) estimate, neglecting broader regions of the posterior distribution. Conversely, MC-dropout samples from a single mode, overlooking the possibility of multiple modes in the posterior.

We propose a novel approach using Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) [8] to quantify the uncertainty of GNNs for the molecular property prediction task. Specifically, we incorporate a cyclical learning rate, which combines the strengths of both sampling strategies and has been successfully utilized in the medical imaging domain [9]. We benchmark the proposed SGHMC method against Deep ensembles and MC-dropout on molecular property prediction tasks. Specifically, we evaluate the models with three different metrics: Error analysis, calibration, and sharpness.

6.2. METHODS

To quantify the uncertainty of GNNs in molecular property prediction tasks, the aim is to transform a single prediction (aleatoric uncertainty) and a deterministic weight (epistemic uncertainty) into a distribution, respectively [10]. For aleatoric uncertainty, the mean-variance estimation is often considered the most practical approach. The mean-variance approach splits the last layers of neural networks to predict both the mean $\mu(x)$ and variance $\sigma^2(x)$ of a Gaussian distribution $\mathcal{N}(y;\mu(x),\sigma^2(x))$. Therefore, during inference time, the test case is evaluated as

$$y_i = \mu(x_i) + \epsilon(x_i), \quad \epsilon \sim \mathcal{N}\big(0,\sigma^2(x_i)\big).$$

Additionally, the corresponding objective function (Eq. 6.1) becomes minimizing the Gaussian negative log-likelihood (NLL) loss function:

$$\mathrm{NLL}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{2\sigma^2_{\mathbf{w}}(x_i)}\big(y_i-\mu_{\mathbf{w}}(x_i)\big)^2 + \frac{1}{2}\log\big(2\pi\sigma^2_{\mathbf{w}}(x_i)\big)\right] \tag{6.1}$$

Moreover, the Bayesian approach is used to get the epistemic uncertainty, which essentially approximates the posterior distribution $p(w \mid D) \propto p(w)p(D \mid w)$. Therefore, during the inference time on a test case: x_i (Eq. 6.2):

$$p(y_i \mid x_i, D) = \int_w p(w \mid D)\, p(y_i \mid x_i, w)\, dw \tag{6.2}$$

However, in reality, the posterior distribution and the integral are intractable. Therefore, they are commonly approximated with MC integration over samples w_i drawn from an approximate posterior distribution $q(w)$ (Eq. 6.3):

$$p(y_i \mid x_i, D) = \frac{1}{M}\sum_{j=1}^{M} p(y_i \mid x_i, w_j), \qquad w_j \sim q(w) \tag{6.3}$$

6.2.1. SGHMC

The proposed SGHMC algorithm samples from the posterior distribution based on the Hamiltonian equation of motion. The Hamiltonian function is given by Equation 6.4:

$$H(q,r) = U(q) + K(r) \tag{6.4}$$

where an object moves on a potential energy surface $U(q)$ with kinetic energy $K(r)$ where q represents the positions and r represents the momenta. Using the canonical distribution (probability of the states based on the energy), the joint probability distribution over positions and momenta can be defined as Equation 6.5:

$$p(q,r) \propto e^{-H(q,r)} = e^{-U(q)}\, e^{-K(r)} \tag{6.5}$$

When the potential energy $U(q)$ is expressed as the NLL of the model combined with a prior, the canonical distribution with respect to $U(q)$ corresponds to posterior distribution of the model parameters w. Minimizing $U(w)$ is equivalent to finding the mode of the posterior distribution:

$$U(w) = -\log\big[\, p(w \mid D)\big] = \mathrm{NLL}(w) + \lambda \, \|w\|^2 \tag{6.6}$$

The choice of distribution over momenta r can be arbitrary. Here, we chose a quadratic form for the kinetic energy:

$$K(r) = \frac{1}{2} r^T M^{-1} r \tag{6.7}$$

resulting in Gaussian distribution over r. The mass matrix M, typically set to the identity matrix for simplification [11]. Therefore, the dynamics in the end are governed by equations:

$$\frac{dw}{dt} = M^{-1} r \, dt \tag{6.8}$$

$$\frac{dr}{dt} = -\nabla U(w) \, dt \tag{6.9}$$

Additionally, to address the computational challenge of evaluating the full gradient $\nabla U(w)$ in large datasets, SGHMC uses a noisy gradient estimate $\nabla \tilde{U}(w)$ based on mini-batches. The introduced noise can be mitigated by a friction term A in the momentum update:

$$r_i = r_{i-1} - \nabla \tilde{U}(w_{i+1}) \, h - A \, r_{i-1} h + \sqrt{2AhT} \, \mathcal{N}(0, I) \tag{6.10}$$

This formulation is similar to SGD with momentum but includes an additional Gaussian noise term, controlled by temperature T. The temperature modulates the sensitivity of the posterior distribution to the potential energy: $T = 1$ yields the standard Bayesian posterior, $T < 1$ (cold posterior) increases sensitivity to low-loss regions, while $T > 1$ broadens exploration at the potential cost of accuracy.

6.2.2. EVALUATION METRIC

The quality of uncertainty estimates in predictive models can be assessed using three key metrics: error analysis, calibration, and sharpness. For the metric error analysis, we utilize the root mean square error (RMSE) and the coefficient of determination (R^2).

Calibration methods assess how well predicted uncertainties align with observed errors. Here, we utilize two metrics: NLL and sum of squared errors (SSE). Rather than only using the NLL as a loss function during training, it is also used as a quantitative measure to assess the quality of the predictive distribution after

training. SSE measures the total squared error between the predicted quantiles and the observed empirical quantiles. If the SSE is small, it means that the predictions of the models are well-calibrated with respect to the observed data.

Sharpness measures the informativeness of uncertainty estimates with root mean predicted variance (RMV) and coefficient of variation C_v. Specifically, RMV is denoted as:

$$\text{RMV} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\widehat{\sigma_i^2}} \tag{6.11}$$

where $\widehat{\sigma_i^2}$ is predicted variance or uncertainty. A low RMV indicates the model on average, predicts low uncertainty and thus low expected error. Additionally, C_v is denoted as:

$$C_v = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{\sigma}_i - \sigma_*\right)}}{\sigma_*} \tag{6.12}$$

where $\hat{\sigma}_i$ is the predicted standard deviation (uncertainty) of instance i and σ_* is the mean predicted standard deviation. A high C_v suggests significant dispersion (heteroscedasticity), indicating that uncertainty estimates are highly dependent on the input. In contrast, a C_v of zero indicates constant (homoscedastic) uncertainty, meaning the estimates are uninformative and do not vary with the input.

6.2.3. EXPERIMENTAL DESIGN

The dataset is the QM9 benchmark [12], consisting of 132,481 small organic molecules (CHONF) with 13 target properties, evaluated at the B3LYP/6-31G(2df,p) level. An 80:10:10 train/validation/test split was applied, using heat capacity at 298.15 K as the target property. The Directed Message Passing Neural Network (D-MPNN) is used as a base model implemented in Chemprop [13], with a mean-variance output and heteroscedastic noise model. The models contain three message passing layers and two feed-forward layers, each with a hidden size of 300.

All uncertainty methods were optimized using the Gaussian NLL loss function with a regularization parameter of $\lambda = 0.0001$. The initial learning rate was set to 0.01, with an exponential decay ($\gamma = 0.95$) and a batch size of 50. MC-Dropout models are trained for 216 epochs with the SGD optimizer and a dropout rate of $p = 0.1$, and 24 predictions are collected for the ensemble with different dropouts. Deep Ensemble models are trained for 650 epochs with the SGD optimizer (momentum $m = 0.9$), with three ensemble predictions from different models. Parallel-SGHMC models are trained for 650 epochs with SGHMC (momentum $m = 0.9$), with noise injected at $T = 0.01$ after 60% of training. Posterior

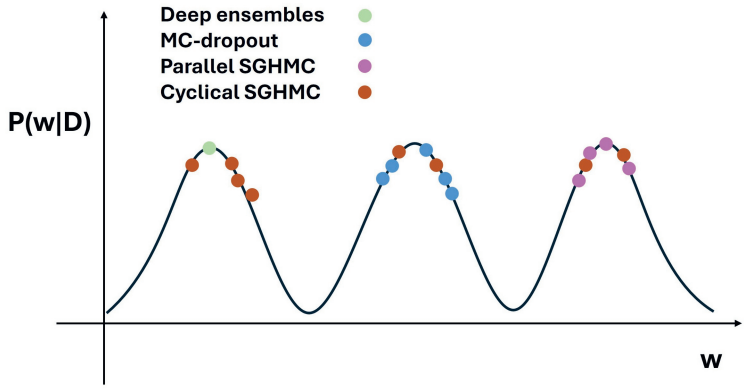


Figure 6.1: Conceptual representation of the different posterior sampling strategies for one training process employed by Deep ensembles, MC-dropout, parallel-SGHMC, and cyclical-SGHMC

samples were collected every ten epochs after noise injection. 24 ensemble predictions are collected from posterior samples. To explore multiple local modes of the posterior, we additionally propose Cyclical-SGHMC model. It uses four cycles within a 650-epoch budget, with a reset learning rate of 0.00025 after the first cycle. Noise was injected with $T = 0.01$ after 60% of each cycle. Similarly, 24 ensemble predictions are collected from posterior samples. All simulations ran on a Windows server with a 3.5 GHz 24-core Intel(R) Xeon(R) W-2265 CPU, an NVIDIA GeForce RTX 3090 GPU, and 64 GB of memory.

Figure 6.1 provides a conceptual representation of various posterior sampling strategies employed during a single training process. The x-axis represents the model parameter space $\mathbf{w}$, while the y-axis shows the posterior probability distribution $P(\mathbf{w} \mid D)$ given the data D. The black curve represents the posterior distribution with multiple modes. Deep Ensembles typically sample from a single local mode within a training cycle, limiting their ability to explore the broader posterior distribution. In contrast, both MC-Dropout and parallel-SGHMC exhibit similar sampling behaviours, generating multiple samples from the local regions surrounding the current mode. Notably, cyclical-SGHMC offers a more robust exploration of the posterior by sampling from multiple regions across different modes, facilitated by its cyclical learning rate, which enables escape from local minima and exploration of diverse posterior landscapes.

6.3. RESULTS AND DISCUSSION

Table 6.1 presents the performance of four uncertainty quantification models - parallel-SGHMC, cyclical-SGHMC, MC-Dropout, and Deep ensemble - on key evaluation metrics, including error, calibration, and sharpness.

In terms of error analysis, SGHMC-based models demonstrate superior performance with respect to predictive error. Specifically, cyclical-SGHMC achieves the lowest RMSE (0.6156) and the highest R^2 (0.9773), indicating that it provides the most accurate predictions. The potential reason is that cyclical-SGHMC samples a richer posterior, capturing parameter uncertainty more effectively than methods like MC-Dropout.

For the calibration category, parallel-SGHMC outperforms the other models with the lowest SSE of 0.0003 and the lowest NLL score of -0.4921. This indicates that its predicted uncertainties are highly aligned with the actual errors, suggesting that this model provides the most reliable uncertainty estimates. Cyclical-SGHMC, with an SSE of 0.0405 and an NLL of -0.3522, also performs well, although its uncertainty estimates are slightly less aligned with the observed errors compared to parallel-SGHMC. By contrast, MC-Dropout and Deep ensemble exhibit poorer calibration performance. Their SSE values of 0.0500 and 0.0012, respectively, and their higher NLL values (0.4678 and 0.4357) suggest that these models tend to overestimate uncertainty. This misalignment between predicted uncertainties and actual errors could lead to less reliable uncertainty estimates in practice, making these models less desirable for applications where accurate uncertainty quantification is important.

In terms of sharpness, parallel-SGHMC performs better than the other models, achieving the lowest RMV of 0.5137 and the highest C_v of 2.0178. This suggests that its uncertainty intervals are narrow and focused, indicating high confidence in its predictions. Cyclical-SGHMC is the next best model with an RMV of 0.6393 and a C_v of 1.6555, providing moderately sharp uncertainty intervals, which reflect a balance between confidence and calibration. On the other hand, MC-Dropout and Deep ensemble exhibit significantly less sharp uncertainty estimates, with RMV values of 0.9167 and 0.7905, respectively. Their lower C_v values (0.9897 for MC-Dropout and 1.0112 for Deep Ensemble) further indicate that these models generate broader uncertainty intervals, reflecting lower confidence in their predictions. While broader intervals can be beneficial in risk-averse or highly uncertain environments, they are less desirable in scenarios where tight confidence bounds are required.

In summary, the parallel-SGHMC model is recommended for tasks requiring accurate predictions, well-calibrated uncertainties, and narrow confidence intervals. Cyclical-SGHMC provides a balanced alternative, while MC-Dropout and Deep ensemble may be more appropriate for scenarios where broader, conservative uncertainty estimates are desired.

Table 6.1: Uncertainty quantification results from four different models. RMSE stands for the root mean square error, R^2 for the coefficient of determination, SSE for the sum of squared errors, NLL for the negative log likelihood, RMV for the root mean predicted variance, and C_v for the coefficient of variation.

Models	Error		Calibration		Sharpness	
	RMSE↓	R^2↑	SSE↓	NLL↓	RMV↓	C_v↑
Parallel-SGHMC	0.6650	0.9735	**0.0003**	**-0.4921**	**0.5137**	**2.0178**
Cyclical-SGHMC	**0.6156**	**0.9773**	0.0405	-0.3522	0.6393	1.6555
MC-dropout	0.7227	0.9687	0.0500	0.4678	0.9167	0.9897
Deep ensemble	0.7399	0.9672	0.0012	0.4357	0.7905	1.0112

6.4. CONCLUSIONS

We proposed a novel approach to perform scalable uncertainty quantification for molecular property prediction with GNNs. Specifically, SGHMC with a cyclical learning rate was researched for scalable uncertainty quantification in molecular property prediction with the D-MPNN architecture with mean variance. A comparative study was performed among SGHMC, Deep Ensemble, and MC-dropout with the QM9 dataset with the heat capacity target property. Our results indicate that the proposed parallel-SGHMC approach outperforms MC-dropout and Deep ensembles in terms of calibration and sharpness. Specifically, parallel-SGHMC reduces the sum of squared errors (SSE) by 99.4% and 75%, respectively. These findings suggest that parallel-SGHMC is particularly suited for applications requiring narrow confidence intervals, where high precision in uncertainty quantification is essential, such as in safety-critical predictions or regulatory settings in pharmaceutical applications. Deep ensemble and MC-dropout may offer advantages in scenarios where broader uncertainty bounds are required, such as in early-stage exploratory research.

REFERENCES

[1] A. M. Schweidtmann, J. G. Rittig, A. König, M. Grohe, A. Mitsos, and M. Dahmen, *Graph neural networks for prediction of fuel ignition quality,* Energy & Fuels **34**, 11395 (2020).

[2] O. Wieder, S. Kohlbacher, M. Kuenemann, A. Garon, P. Ducrot, T. Seidel, and T. Langer, *A compact review of molecular property prediction with graph neural networks,* Drug Discovery Today: Technologies **37**, 1 (2020).

[3] J. G. Rittig and A. Mitsos, *Thermodynamics-consistent graph neural networks,* Chemical Science **15**, 18504 (2024).

[4] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, V. Makarenkov, and S. Nahavandi, *A review of uncertainty*

quantification in deep learning: Techniques, applications and challenges, Information Fusion **76**, 243 (2021).

[5] Y. Gal and Z. Ghahramani, *Dropout as a bayesian approximation: Representing model uncertainty in deep learning,* (2015), 10.48550/ARXIV.1506.02142, arXiv:1506.02142 .

[6] B. Lakshminarayanan, A. Pritzel, and C. Blundell, *Simple and scalable predictive uncertainty estimation using deep ensembles,* (2016), 10.48550/ARXIV.1612.01474, arXiv:1612.01474 .

[7] F. Wang, Y. Liu, K. Liu, Y. Wang, S. Medya, and P. S. Yu, *Uncertainty in graph neural networks: A survey,* Transactions on Machine Learning Research (11/2024) (2024), 10.48550/ARXIV.2403.07185, arXiv:2403.07185 .

[8] T. Chen, E. B. Fox, and C. Guestrin, *Stochastic gradient hamiltonian monte carlo,* (2014), 10.48550/ARXIV.1402.4102, arXiv:1402.4102 .

[9] Y. Zhao, C. Yang, A. Schweidtmann, and Q. Tao, *Efficient bayesian uncertainty estimation for nnu-net,* in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2022* (Springer Nature Switzerland, 2022) pp. 535–544.

[10] J. Busk, P. Bjørn Jørgensen, A. Bhowmik, M. N. Schmidt, O. Winther, and T. Vegge, *Calibrated uncertainty for molecular property prediction using ensembles of message passing neural networks,* Machine Learning: Science and Technology **3**, 015012 (2021).

[11] S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, *Handbook of Markov Chain Monte Carlo* (Chapman and Hall/CRC, 2011).

[12] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld, *Quantum chemistry structures and properties of 134 kilo molecules,* Scientific Data **1** (2014), 10.1038/sdata.2014.22.

[13] E. Heid, K. P. Greenman, Y. Chung, S.-C. Li, D. E. Graff, F. H. Vermeire, H. Wu, W. H. Green, and C. J. McGill, *Chemprop: A machine learning package for chemical property prediction,* Journal of Chemical Information and Modeling **64**, 9 (2023).

IV

GRAPH MACHINE LEARNING FOR PROCESS DESIGN

7

ACCELERATING PROCESS SYNTHESIS WITH REINFORCEMENT LEARNING: TRANSFER LEARNING FROM MULTI-FIDELITY SIMULATIONS AND VARIATIONAL AUTOENCODERS

Reinforcement learning has shown some success in automating process design by integrating data-driven models that interact with process simulators to learn to build process flowsheets iteratively. However, one major challenge in the learning process is that the reinforcement learning agent demands numerous process simulations in rigorous process simulators, thereby requiring long simulation times and expensive computational power. We propose employing transfer learning to enhance the reinforcement learning process in process design. This study examines two transfer learning strategies: (i) transferring knowledge from shortcut process simulators to rigorous simulators, and (ii) transferring knowledge from process variational autoencoders (VAEs). Our findings reveal that appropriate transfer learning can significantly improve both learning efficiency and convergence scores. However, transfer learning can also negatively impact the learning process when there are substantial discrepancies in the decision range and reward function. This suggests that pre-trained process data should match the complexity of the fine-tuning task.

This chapter has been published as "Accelerating process synthesis with reinforcement learning: Transfer learning from multi-fidelity simulations and variational autoencoders" by Gao, Q., Yang, H., Theisen, M. F., & Schweidtmann, A. M. Computers and Chemical Engineering, 201(10919), 109192, (2025)

7.1. INTRODUCTION

Conceptual process design is a complex task executed by engineers. Classical conceptual process design methods are either manual works that demand long development times or superstructure methods that are limited to the pre-defined superstructures and difficult to solve the resulting mixed-integer nonlinear optimization problems (MINLPs) [1]. Additionally, artificial intelligence (AI) has also been leveraged in conceptual process design. Notably, a number of expert systems [2, 3] were proposed in the first wave of AI where experts integrate domain experience and facts into a knowledge base to perform reasoning for design problems. However, those expert systems were difficult to maintain and extend, which hindered further developments in conceptual process design.

Recently, reinforcement learning (RL), a subclass of machine learning (ML), has demonstrated excellent performance in solving complicated sequential decision-making problems such as gaming [4, 5], process control [6], and molecular design [7]. RL is a computational approach in which an agent iteratively interacts with its environment to sequentially make decisions and achieve its objectives [8]. Several initial first steps have been made towards applying RL in conceptual process design, as summarized in our recent review paper [9]. Perera et al. [10] first utilized an RL agent as a supporter for suggesting dispatch strategies for energy systems design based on the pre-defined structure. Khan and Lapkin [11] utilized RL to search the optimal flowsheet with a hydrogen production process case study, where a value-based agent was used to estimate the value for the next processing step based on the intimidate flowsheet. Furthermore, Midgley [12] leveraged a soft-actor-critic RL agent to design a separation process for non-azeotropic mixtures. In their approach, the agent first chooses whether to add a new distillation column and then chooses the corresponding continuous operation variables, and the flowsheet is simulated in the open-source process simulator COCO. Additionally, Göttl et al. [13] represented a full flowsheet in a matrix for two competing players as state space and proposed hierarchical reinforcement learning (HRL) to design more advanced processes including recycles. HRL is an approach that decomposes a complex task into simpler subtasks. For example, in the context of process design, the agent first determines an open stream to add a unit operation, then the type of unit operation, then design variables, and, finally, operating variables. Khan and Lapkin [14] also implemented HRL where a high-level agent chooses a sub-objective of the process section and a low-level agent chooses unit types and corresponding control variables within the section. Moreover, Plathottam et al. [15] introduce RL in a solvent extraction process design by optimizing design variables for predefined flowsheets. In a recent publication, Stops et al. [16] introduced a graph-based representation of the process, which more naturally captures the flowsheet topologies, and utilized the HRL agent to design processes.

Reynoso-Donzelli and Ricardez-Sandoval [17, 18] further implemented a masked RL agent which allows the exclusion of infeasible or irrelevant actions based on process constraints, ensuring that the RL agent focuses only on viable process decisions. Furthermore, several studies deployed hybrid approaches within the RL framework. For example, Tan et al. [19] combinned RL with mathematical programming to perform the design of heat exchanger networks. Also, Göttl et al. [20] utilized RL for the discrete decisions alongside a genetic algorithm for the continuous decisions for hybrid action spaces, which is illustrated by two real-world use-cases: Synthesis of chemical process flowsheets and design of optical multi-layer films.

Although there are some successful applications for RL in process design, one major remaining challenge is that the RL training process is trial-and-error-based. Thereby, the learning process typically requires a large number of process simulations, which demands expensive computational power and long simulation times. Therefore, to mitigate this problem, most works previously deployed shortcut process simulation methods to efficiently simulate the process, which may potentially lead to inaccurate results. Recently, Kalmthout et al. [21] leveraged a rigorous process simulator (Aspen Plus) to perform RL in process design tasks. Recently, Göttl et al. [22] demonstrated a generalized RL framework that can adapt to different chemical systems during the inference time without retraining, which significantly reduces the computational burden associated with RL-based process design. However, the long simulation time and convergence issue in the training phase still hinder further advances.

Transfer learning has been used to accelerate the learning process in many domains such as computer vision [23], and natural language processing [24]. Transfer learning is a method that transfers knowledge from different but relevant domains to the target domain, thereby improving the learning performance [25]. In the context of RL in process design, Wang et al. [26] first trained the RL agent with one case study and then transferred the RL agent to another case study to demonstrate the generalizability of the RL agent. With their following work, Tian et al. [27] further applied RL to a hydrodealkylation example. Additionally, our previous work [28] demonstrates that pretraining RL agents on the shortcut process simulation method and then fine-tuning the transferred agents in the rigorous simulator can not only accelerate the learning process but also improve the convergence score. Therefore, transfer learning has huge potential to mitigate long simulation time problems with rigorous simulators for RL in process design. However, there exist many possible strategies to perform transfer learning. Comparing them and identifying promising ones is still an open research question. Also, the effectiveness of transfer learning is not always guaranteed and it has been observed that transfer learning can bring negative contributions to

the fine-tuning task, namely negative transfer [29]. The effect has not yet been discussed in the context of RL for process design.

We propose and investigate two transfer learning strategies: (i) transfer learning from shortcut simulation methods to a rigorous process simulator and (ii) transfer learning from generative AI models. In the first strategy, we first pre-train the RL agent with shortcut simulation methods and then further fine-tune it with a rigorous process simulator, DWSIM. Specifically, we explore different transfer methods within the RL agent architecture and apply these to an illustrative case study comprising equilibrium reactions, azeotropic separation, and recycles. In the second strategy, we utilize the encoder of a variational autoencoder (VAE), which was pre-trained on flowsheet data, and then further fine-tuned with rigorous process simulator DWSIM.

7.2. RL FOR PROCESS SYNTHESIS

A RL problem can be formulated as a Markov decision process (MDP): $M = \{S, A, T, R\}$ with states $\mathbf{s} \in S$, actions $\mathbf{a} \in A$, the transition function $T : S \times A \times S \rightarrow [0,1]$, and the reward function $R : S \times A \times S \rightarrow \mathbb{R}$. Figure 7.1 shows a framework in the context of RL in process design. The agent learns to design processes by iteratively placing unit operations together with corresponding design/operating variables, simulating the process in the environment, and maximizing the long-term rewards, thereby obtaining optimal processes. Specifically, the states **s** represent the information including flowsheet topology as well as design variables, operating variables, thermodynamic stream data, flowrates, and compositions. The agent, typically consisting of neural networks, takes the current states **s** as input and based on that takes actions **a**. Notably, the action space can be hybrid and hierarchical. A hybrid action space means the actions include discrete choices (e.g., the selection of open streams, or unit operation types) and continuous choices (e.g., the length of a reactor, or operating temperature). A hierarchical action space indicates that the agent can make decisions sequentially. For example, the agent first determines an open stream to add a unit operation, then the type of unit operation, then design variables, and finally operating variables. In chemical processes, the design variables are usually determined during the initial design phase and typically remain fixed throughout the operation. While operating variables can be adjusted during operation. After a new unit operation is added, the new flowsheet is simulated in an environment, which is typically a process simulation software. Subsequently, the environment sends a numerical reward $\mathbb{R}$ to the agent, which is the corresponding objective for optimization such as net present value. Ultimately, the agent learns to design optimal flowsheets by repeating the whole steps and maximizing the cumulative rewards.

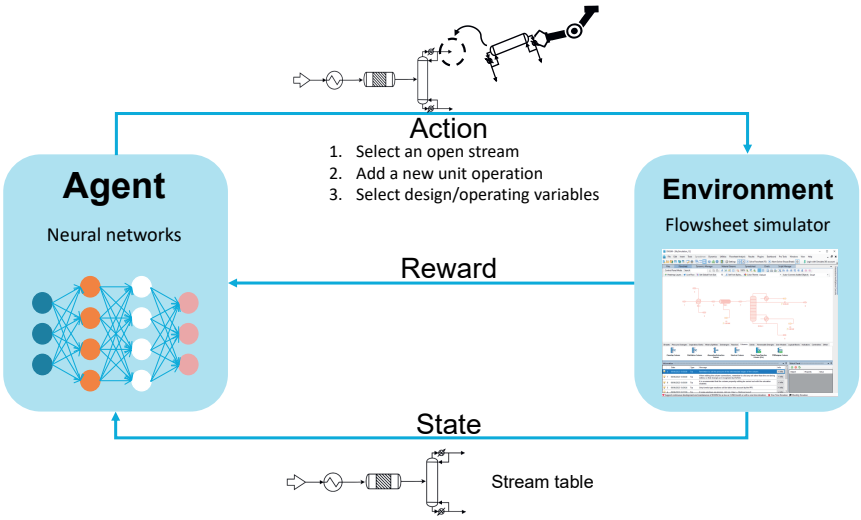


Figure 7.1: General framework of RL in process design

7.2.1. INFORMATION REPRESENTATION

Chemical processes contain various information such as topology, design variables, operating variables, thermodynamic stream data, flowrates, and compositions. Presenting this information in a meaningful way is critical for the learning process of the RL agent. In this work, we present the flowsheets as directed homogeneous graphs as Figure 7.2 shows. Within the process graph, the feeds, unit operations, and products are denoted as nodes, namely vertices $v \in V$, and streams are denoted by directed edges $e_{vw} \in E$ linking two nodes v and w. Notably, each node and edge is associated with one node feature vector $\mathbf{f}^v \in F^V$ and edge feature vector $\mathbf{f}^{e_{vw}} \in F^E$, respectively. The node feature vector typically contains types of unit operation, corresponding design variables, and operating variables. The edge feature vector includes thermodynamic stream data, compositions, and flowrates. Additionally, in the uncompleted flowsheet, open streams are linked to "undefined" nodes because open edges are prohibited in graph representations. These undefined nodes serve as temporary placeholders for future unit operations. Essentially, incorporating a new unit operation involves substituting an "undefined" node with a specific type of unit operation node.

7.2.2. AGENT ARCHITECTURE

In this section, we introduce the proposed agent architecture, as Figure 7.3 shows. The architecture is actor-critic based and consists of three parts: A graph encoder, hierarchical actor networks, and critic networks. First, a graph neural network

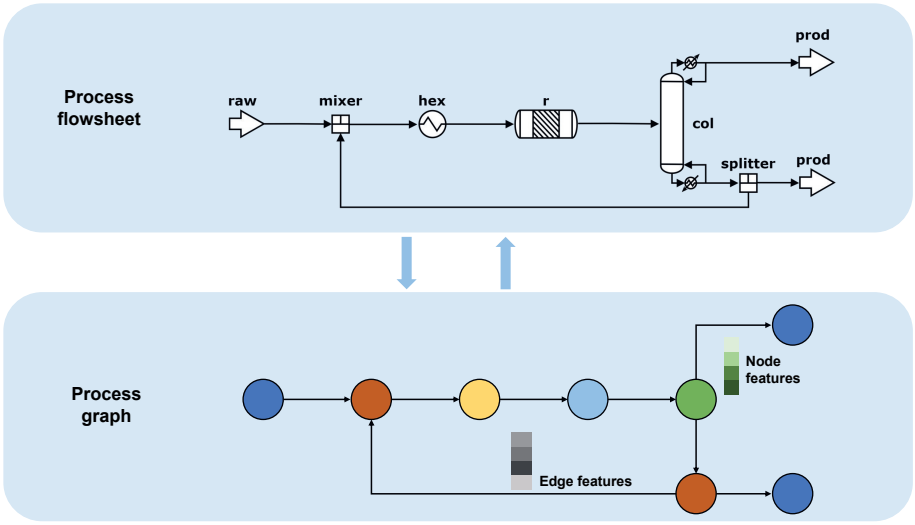


Figure 7.2: One example of process graph. Feeds, unit operations, and products are represented as nodes which are associated with node features. Streams are represented as edges which are also associated with corresponding edge features.

(GNN) is utilized to encode flowsheet graphs into a flowsheet fingerprint. The actor then takes actions at three levels: (1) select an open stream, (2) select a unit operation and (3) select a design/operating variable. The critic networks evaluate the value of the flowsheet fingerprint to optimize the reward functions. In Section 7.2.2, Section 7.2.2, and Section 7.2.2, we will introduce the graph encoder, hierarchical actor networks, and the critic network, respectively.

GRAPH ENCODER

The flowsheet graphs are processed by a GNN, as illustrated in Figure 7.4. The architecture of the proposed GNN consists of two primary phases: message passing and readout. During the message passing phase, graph convolutional networks (GCNs) are utilized to update the node embeddings for the flowsheet graph. Specifically, Figure 7.5 illustrates the message passing step for a node represented in red. Initially, the message function **m** combines the information from neighboring nodes (colored yellow) and their connecting edges into a message. Subsequently, this message is utilized to update the feature of the targeted node using an update function **u**, such as the sum function. This update process is iteratively applied to each node in every layer $l \in L$ within a GCN, allowing nodes to cumulatively gather information from their neighbors up to a distance determined by the number of layers L. Afterward, in the readout phase, the structure information learned during the message passing phase for each node is aggregated into a flowsheet

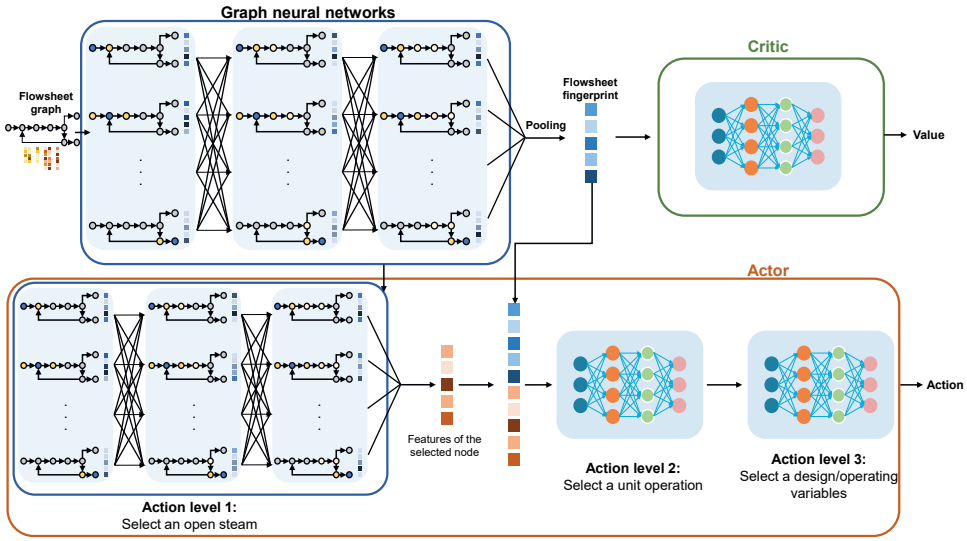


Figure 7.3: The example agent architecture.

fingerprint by a pooling function.

HIERARCHICAL ACTOR NETWORKS

The architecture of actor networks is primarily shaped by a hierarchical and hybrid action space, as illustrated in Figure 7.6. This framework encompasses three decision-making levels: (i) selecting an open stream, (ii) choosing a specific unit operation, and (iii) identifying a design or operating variable. The initial two levels involve making discrete choices, whereas the final level is continuous, which together are referred to as a hybrid action space. In the first level, the agent selects an open stream which marks the location for subsequent expansion within the flowsheet.

In the second level, the agent selects the type of unit operation to be added, which could be a heat exchanger, a reactor, a distillation column, or even a recycling process through the introduction of an additional splitter and mixer. This step also includes the option of choosing an open stream as a product, effectively terminating the expansion for that particular pathway. The decisions in the third level are directly tied to those in the previous level. For instance, upon the addition of a reactor, the agent is tasked with determining a specific design variable, such as the reactor length. For simplicity, only one variable is selected for each unit operation. It is important to note that within the current implementation, the recycle stream can be integrated solely into the feed stream.

The implementation of actor networks echoes the hierarchical and hybrid

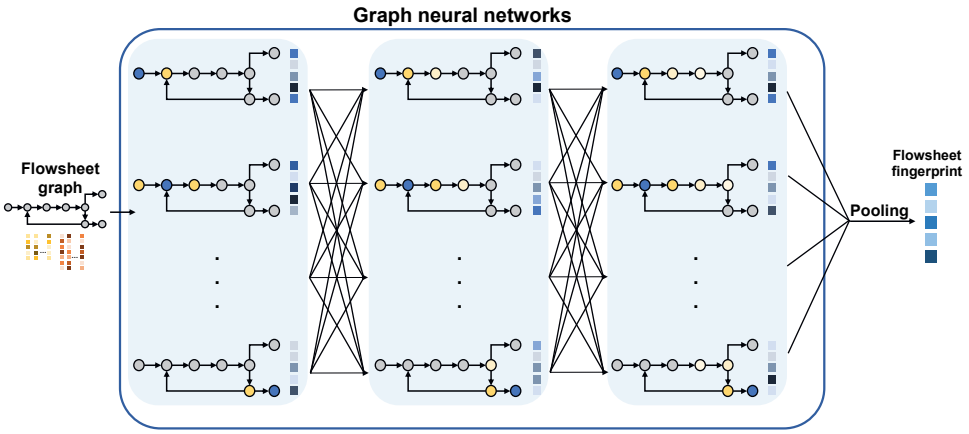


Figure 7.4: The example architecture of the graph encoder, which is adapted from Schweidtmann et al. [30]

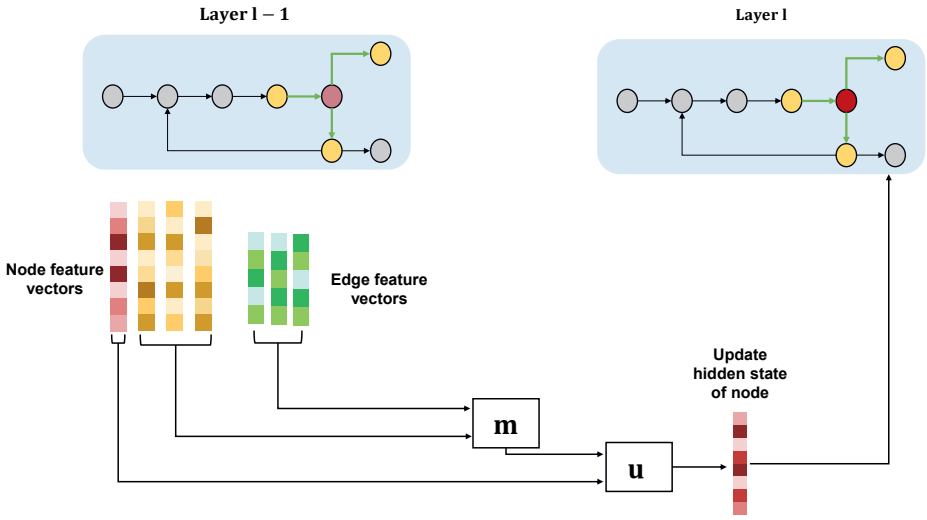


Figure 7.5: An overview of node message passing in graph convolution layers.

action space, as depicted in Figure 7.3. In action level 1, the updated graph (before the readout phase) in the graph encoder is further processed by two additional GCN layers. Within these two GCN layers, all node features are compressed to a single dimension, similar to the approach used in node classification tasks. Furthermore, we apply a zero mask vector to effectively eliminate the selection probability of all nodes, with the exception of those categorized as "undefined." This ensures that nodes corresponding to feed and unit operations are excluded from selection. This selected node feature is concatenated with the flowsheet fingerprint, forming the input for subsequent action levels. In action level 2, a Multilayer Perceptron (MLP) calculates probabilities for each possible unit operation. For action level 3, each unit operation is associated with its own MLP, which predicts two parameters, α and β, for a beta distribution $B(\alpha, \beta)$ A continuous decision regarding the respective design/operating variable is then made based on this beta distribution.

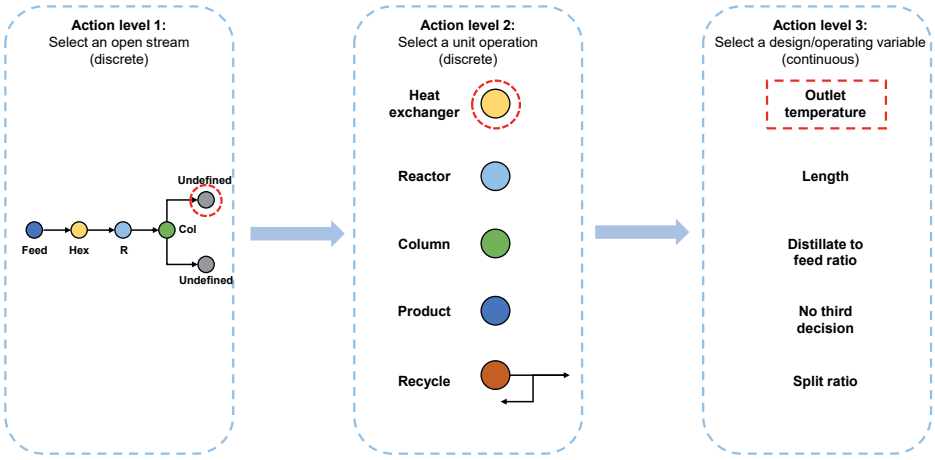


Figure 7.6: An overview of node message passing in graph convolution layers.

CRITIC NETWORKS

The critic network estimates the value of the current state, providing crucial feedback to the actor network. [8]. This value estimates the total expected rewards from the current state to the end of an episode under the current policy. This feedback helps the actor adjust its policy to maximize long-term rewards. Specifically, as shown in Figure 7.3, in our framework we calculate a generalized advantage estimation $\hat{A}$, which assesses whether actions taken yield better or

worse outcomes than predicted, which is crucial for computing the critic network's loss function.

7.3. CASE STUDY

The proposed framework is illustrated through a case study centered on a hypothetical reaction synthesis process and its according separation. This process operates within a chemical system comprising four compounds: A, B, C, and D. Additionally, we consider a scenario in which the resulting mixtures are assumed to behave ideally and are separated exclusively through distillation. In all simulations, the initial conditions feature a feed composed of an equimolar binary mixture of A and B. The molar flow rate is set to 100 mol s^{-1}, with the feed temperature maintained at 300 K.

7.3.1. ENVIRONMENT

Two different environment settings are utilized in this study: Short-cut process models and DWSIM. The short-cut process methods for pre-training are illustrated in our previous work [16]. Here, we introduce the DWSIM process simulator as an RL environment. As introduced in Section 7.2.2, the agent can choose between reactors, distillation columns, and heat exchangers as unit operations. Besides, the agent can also decide to add recycles or claim open streams as products. The types of unit operations and the corresponding design variables are defined below.

REACTOR

A plug flow reactor (PFR) is deployed to convert reactants to the desired products. Specifically, the following reversible reaction takes place as Equation 7.1 shows and C is the desired product.

$$A + B \rightleftharpoons C + D \tag{7.1}$$

To further enhance the reaction engineering challenge, we assumed the reaction kinetics of our hypothetical reaction to follow those of Xu and Chuang on the esterification of acetic acid with methanol to produce methyl acetate [31] However, it is important to note that while we have adopted their kinetic data, our case study differs from their experimental setup. Specifically, we have simplified the model by not considering catalyst loading and simulating the reactor isothermally which indicates the constant reaction rates. We note that using a PFR was partly an arbitrary choice for this case study, as it is a common reactor type. Additionally, the reactor cross-sectional area is determined by $N/10$, where N is the inlet molar flow. The design variable is the reactor length, which will be determined by the agent in the third-level continuous decision process. The range is from 3 to 30 m.

DISTILLATION COLUMN

A distillation column is provided in the environment to separate product C from the quaternary system. Rigorous distillation columns are used instead of shortcut columns in the previous work [16] to account for more realistic factors such as varying stage conditions. The rigorous column models provide multiple possible choices of design parameters, from which the distillate to feed ratio (D/F) is selected as the third-level decision. Other adjustable parameters such as the number of stages and reflux ratio are set as fixed values (35 and 1.5, respectively). The D/F ratio can range from 0.3 to 0.7.

HEAT EXCHANGER

For temperature changes, we provide the agent with the DWSIM heater model. In the proposed framework, the heat exchanger is simulated based on the outlet temperature, which is used as a design variable determined by third-level decisions. A temperature range from 278.15 to 330.05 K is defined, where the upper limit refers to the lowest boiling point of the components, which is product C.

RECYCLE

The recycling action consists of additional units that include a splitter and a mixer. Firstly, the process stream is split into a recycle stream and a purge stream. Secondly, the recycled stream is merged with the selected feed using a mixer. Thereby, the split ratio is the third-level decision of the agent, which lies in the range of 0.1 to 0.9. In the current setting, the recycle destination is always the feed stream. In principle, it is possible to predefine the destination of a recycle stream within our framework. For instance, one might specify that the recycle stream can only be directed to the inlet before the reactor. When the agent selects a recycle action, the RL environment can then check whether a reactor exists. If so, the recycle stream is placed at the reactor inlet; if not, the action is deemed invalid, and a penalty is given. However, in a more realistic scenario, the RL agent should have the flexibility to determine the optimal recycle location. Future work could address this by introducing an additional action level 4, leveraging GNNs for edge (stream) classification, similar to action level 1 (node classification). When the RL agent selects a recycle action, the action level 4 GNN will predict the most suitable recycle destination.

7.3.2. REWARD

We calculate the reward according to Equation 7.2 when a flowsheet is completed.

$$r = \sum P_{\text{products}} - \sum C_{\text{feed}} - \sum \left(C_{\text{operation}} + 0.15 * C_{\text{invest}} \right)_{\text{units}} \tag{7.2}$$

where P_{products} is the revenue of the sold product (Seider et al., 2008), C_{feed} is the costs of feeds, $C_{\text{operation}}$ is the operation costs (Smith, 2016) and C_{invest} is the total capital investment which is multiplied by factor 0.15 (Seider et al., 2008). In the case of negative rewards, a reduction factor of 10 is applied to encourage the exploration of the design space. Additionally, a reward of 0 € is given when the incomplete flowsheets can converge after every single action because the economic value is difficult to assess for an incomplete flowsheet. Whenever the agent fails the simulation by taking infeasible actions, the episode will be terminated immediately, and a negative reward -10M € is given.

7.3.3. CONSTRAINTS

In our framework, we incorporate constraints at three levels: variable bounds, soft constraints, and masking. Hard constraints are a promising future work. First, we apply variable bounds (aka design constraints) to define the ranges of feasible values for each decision. For example, we predetermine the allowable PFR length so the RL agent can only select feasible lengths. Second, we introduce soft constraints that penalize infeasible actions through the reward function without strictly forbidding them. For instance, if an action leads to non-convergence, the agent incurs a large negative reward, discouraging it from selecting similar actions again. A temperature constraint on the PFR could be also handled through a soft constraint method. Finally, we also employ an action-masking strategy. For example, in action level 1, when selecting an open stream, we apply action masking to prevent the selection of "illegal" streams, ensuring that only valid open streams are chosen. However, the consideration of nonlinear constraints is an active field of research in RL. There exist a number of methods that could be integrated into our framework as a future work. For example constrained RL algorithms [32] and safe exploration techniques [33] have been successfully applied in control settings to handle constraints, and could similarly be leveraged for process design tasks in future work.

7.3.4. TRAINING PROCESS

The method described is implemented using Python 3.9, with the training scheme based on the Proximal Policy Optimization (PPO) algorithm developed by OpenAI [34]. This approach involves several epochs of minibatch updates, where each minibatch is formed by sampling transition tuples that have been saved in memory. The agent's networks are updated through gradient descent. The final loss is the weighted sum of each individual actor network, their entropies, and the critic's loss, ensuring a balanced update mechanism that considers both exploration (via entropy) and exploitation (via actor and critic losses).

The agent's decision-making capabilities are enhanced throughout the train-

ing process by adjusting its weights and biases. In our training setup, the agent's parameters are updated every 20 steps. At each update interval, the agent executes a batch loss computation using 30 randomly selected samples from its memory, subsequently updating its parameters through the backpropagation algorithm.

The experiments are performed on a Windows server with a 3.5 GHz 24 cores Intel(R) Xeon(R) W-2265 CPU, NVIDIA GeForce RTX 3090 GPU, and 64 GB memory. The hyperparameters can be found in Electronic Supplementary Information (ESI).

7.3.5. TRANSFER LEARNING

Transfer learning is a machine learning technique where a model developed for one task is reused as the starting point for a model on a second task, incorporating pre-existing knowledge to improve performance or reduce training time. We propose to utilize two different transfer learning strategies as Figure 7.7 illustrates. Table 7.1 depicts detailed transferred components of the models for two strategies.

In the first transfer learning strategy ($T1$), we pre-train the agent using a shortcut process models from our prior research [16], followed by fine-tuning with the rigorous DWSIM simulator on the same case study. Throughout both phases, the agent maintains an identical architecture. We explore four distinct experiments by transferring various combinations of model components groups E1 to E5.

In the second transfer learning strategy ($T2$), a graph VAE is developed to learn from syntactical process data and actual process topology data. The actual process topology data comprises 730 flowsheets from simulation files [35], P&ID-like flowsheets without control structure [36], and mined flowsheets from literature and the internet [37]. The synthetic dataset consists of 7952 flowsheets, which are generated based on a Markov chain-like sampling with fixed probabilities and predefined local patterns, such as reaction, thermal separation, or recycling [35]. Only the encoder component of the process VAE is then transferred and fine-tuned using the rigorous process simulator DWSIM (group E6). For implementation details of the VAE, we refer readers to our paper [38].

7.4. RESULTS AND DISCUSSION

In this part, we will present the results of the two different transfer learning strategies separately. Specifically, in Section 7.4.1, we will discuss the results of learning curves from the with and without transfer learning groups. Additionally, Section 7.4.2 presents the flow sheets generated from all the groups.

Table 7.1: Overview of transfer learning models strategies.

Model components	$T1$					$T2$
	E1	E2	E3	E4	E5	E6
Graph encoder	✓	✓	✓	✓	✓	✓
Actor level 1	-	✓	✓	✓	✓	-
Actor level 2	-	-	✓	✓	✓	-
Actor level 3	-	-	-	✓	✓	-
Critic network	-	-	-	-	✓	-

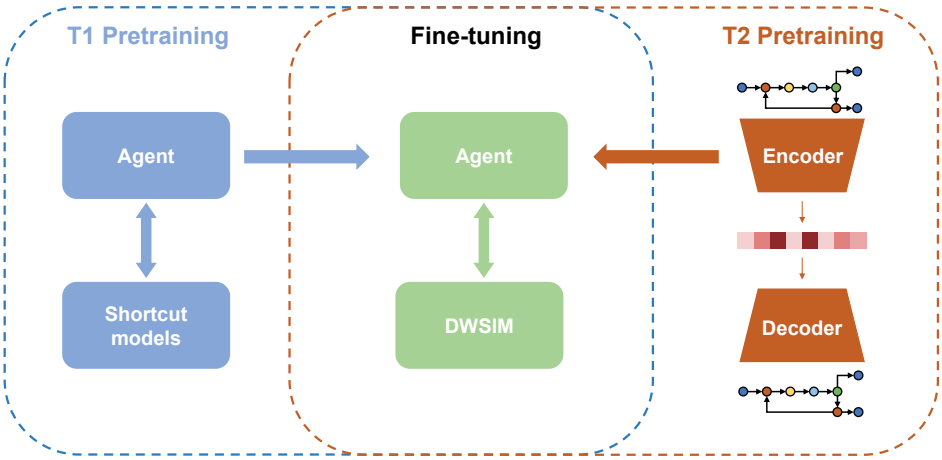


Figure 7.7: Overview of transfer learning. In T1 pretraining, the agent interacts with shortcut models to build foundational knowledge. The agent is then fine-tuned with the rigorous process simulator DWSIM. Concurrently, T2 pretraining involves an encoder-decoder architecture, and then the part of the encoder is transferred and fine-tuned with the rigorous process simulator DWSIM.

7.4.1. LEARNING CURVES

Figure 7.8 and Table 7.2 display the learning curves and convergence scores for the baseline group (C1) and two transfer learning strategies T1 and T2. Every group conducted over 10000 episodes for three individual runs because we observed that the learning curves plateaued and showed no further improvement after 10,000 episodes. The score represents the moving average of the rewards over every 100 episodes. Specifically, the solid line in Figure 7.8 represents the mean of three individual runs and the shaded area represents the standard deviation for each group. The training for the agent without transfer learning took 72 hours over 10000 episodes. For the groups (E1 - E5) in the transfer learning strategies T1, the pre-training took 2 hours over 10000 episodes and the further training took 72 hours over 10000 episodes. Furthermore, for the group (E6) in the transfer learning strategies T2, the pre-training took 6 hours and the further fine-tuning process took 72 hours over 10000 episodes. Table 7.2 provides an overview of the convergence plateaus for groups with and without transfer learning. The convergence score, measured in millions of euros (ME), and the number of episodes required to reach various profitability thresholds (0 M€, 20 M€, 40 M€, 60 M€) are reported.

Upon learning convergence, all agents demonstrate the capacity to create feasible flowsheets, each achieving average scores exceeding 60 million euros. The baseline group without transfer learning (C1) has a convergence score of 60.24 ± 3.25 M€. In contrast, groups E1 to E5, which incorporate transfer learning, all achieve higher convergence scores. For example, E1, which transfers only the graph encoder, improves the convergence score by approximately 8.1% compared with C1. Remarkably, E5 with all transferred agent architectures has the highest convergence score 66.81 ± 2.84 M€, representing an improvement of 11.0% over C1. The results demonstrate that agents can successfully transfer the information learned in the shortcut environment to enhance performance when fine-tuning in rigorous environments. Furthermore, we utilize reaching episodes on certain thresholds to measure and compare the training efficiency and speed. For group C1, reaching the 0 M€, 20 M€, 40 M€, and 60 M€ milestones required 126, 1267, 2413, and 3975 episodes, respectively. Remarkably, groups E3, E4, and E5 demonstrate a significantly quicker learning process than C1. For example, all three groups can generate feasible flowsheets (> 0 M€) in the first episode, indicating the transferred information from the shortcut column contributes to making feasible action in the very early stage when fine-tuning in rigorous simulation. Notably, E3 in the first episode can reach the 40 M€ threshold which decreases the training episode time by 99.92%. Additionally, E5 with transferring all components substantially boosts learning velocity with only 1436 episodes for 60 M€ milestones, which decreases the training episode time by 63.88% than C1. On the other hand,

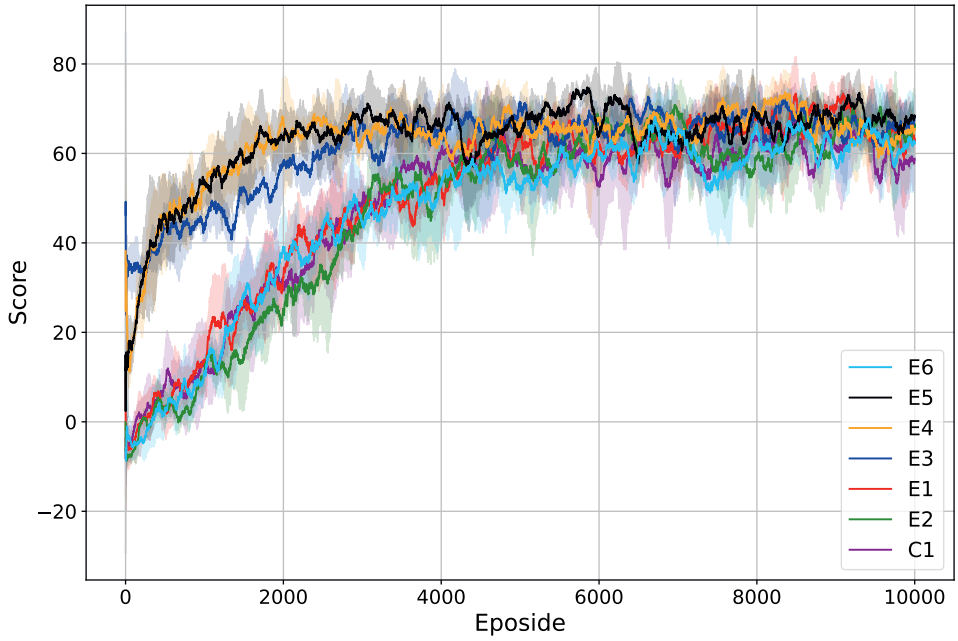


Figure 7.8: Learning curves of all groups, smoothed scores plotted with standard deviations. C1 represents the score from the agents without transfer learning. E2-E5 represent the scores from the agent with transfer strategy T1. E6 represents the score from the agent with transfer strategy T3. The detailed transfer learning strategies depict in Table 7.1.

transfer learning strategies E1 and E2 led to less improved or even slower learning processes. On the other hand, transfer learning strategies E1 and E2 resulted in less improvement and even slower learning processes. Specifically, both groups required 4303 and 4276 episodes, respectively, to reach the 60 million euro milestone, increasing the training episodes by 8.22% and 7.55%. This suggests that for E1 and E2, the agents may not efficiently utilize the state representations and initial policy guidance provided by the pre-trained encoder and first-level actor, which hinders comprehensive policy optimization and efficient exploitation of the learned state representations. In contrast, transferring additional actor levels and the critic, as seen in E3, E4, and E5, helps to better refine policies and value estimations, leading to faster convergence and higher overall performance.

The learning curve of E6 with the transferred VAE encoder demonstrates a similar performance compared with C1. Quantitatively, the convergence score of E6 increases by 1.46% compared to C1. However, E6 requires significantly more episodes (333) to reach 0 M€, and ultimately 5505 episodes to reach 60 M€, indicating a less efficient learning process. This discrepancy may arise from the misalignment between the pre-training process data and the target process in RL. The pre-training process data consists of 730 flowsheet topologies from different resources [37, 39], which are comparatively larger than the target process in RL and typically lack steam information. The processes used for pretrapping the VAE are further a diverse range of processes with a broader range of chemicals, unit operations, and topologies. Therefore, the encoder from VAE during the pertaining process potentially yields mismatched learning semantics between the pre-training phase and fine-tuning phase. The mismatched semantics preserved in the encoder hinder further learning and converging in the fine-tuning within the target RL task.

In summary, transfer learning from a shortcut process simulator to a rigorous simulator significantly enhances the learning performance of agents, as evidenced by all the higher convergence scores of groups E1 to E5. The varying degrees of improvement in learning efficiency between groups may also indicate the differential impact of the types of knowledge transferred, the strategies employed, and the adaptation to the new environment. In terms of learning curves, overall, group E5 demonstrates the highest convergence scores and the most efficient learning process. On the other hand, the results of transfer learning from a pre-trained VAE highlight the challenges of applying transfer learning in RL for process design, emphasizing the importance of aligning pre-training tasks closely with the target RL tasks to achieve meaningful performance improvements.

Table 7.2: Overview of convergence plateaus of with and without transfer learning groups

Group	Convergence score (M€)	Reaching episodes			
		0 M€	20 M€	40 M€	60 M€
C1	60.24 ± 3.25	127	1268	2414	3976
E1	65.12 ± 3.35	**1**	1063	2136	4303
E2	63.12 ± 3.66	201	1599	2712	4276
E3	66.59 ± 2.44	**1**	**1**	**1**	2373
E4	66.62 ± 3.03	**1**	**1**	324	1503
E5	**66.81 ± 2.84**	**1**	113	311	**1436**
E6	61.13 ± 3.45	333	1261	2125	5505

7.4.2. GENERATED FLOWSHEETS

In this part, we discuss the generated flowsheets from all groups. Figure 7.9 shows the generated flowsheet with the highest score obtained from the pre-training phase of the shortcut model, which resembles our previous work [16]. Furthermore, Figure 7.10 displays the flowsheet from C1 (250 episodes). Notably, agents without transfer learning tend to create unfeasible and lengthy flowsheets including multiple unnecessary heat exchangers in the early stage. This is due to the random exploration as part of the trial-and-error nature of the RL. With random initialization, agents have an inadequate learning experience and a preference to consistently select heat exchangers because of their relatively low capital cost and ease of convergence. Figure 7.11 visualizes the flowsheet from E3 (1 episode). Notably, with the transferred graph encoder and actor networks 1 and 2, agents can already generate a concise and feasible flowsheet in the first episode compared with C1. This additionally verifies that the agent can leverage the learned information from the shortcut process model and make quick and feasible decisions at a very early stage. Moreover, the decision-making ability of agents is reinforced by accumulated experiences, and thus, better initialization can contribute to long-term outcomes. For example, as Table 7.2 shows, the convergence score of E2 increases by 8.41% than C1.

To delve deeper into the performance of agents from different groups, we discuss the flowsheets with the highest scores during the training phase from different groups. Figure 7.9 illustrates the shortcut-method flowsheet that achieved the highest reward (41.66 M€). The feed stream (F-101) first passes through a plug flow reactor (R-101), and the resulting quaternary mixture is then separated in a distillation column (C-101). The final product, component C, obtains from product stream P-102 at 56.4% purity. Additionally, Figure 7.12 presents the flowsheet

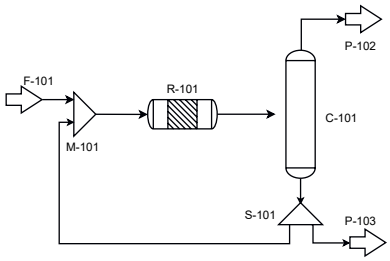


Figure 7.9: The generated flowsheet with the highest score 41.66 M€ obtained from the shortcut model.

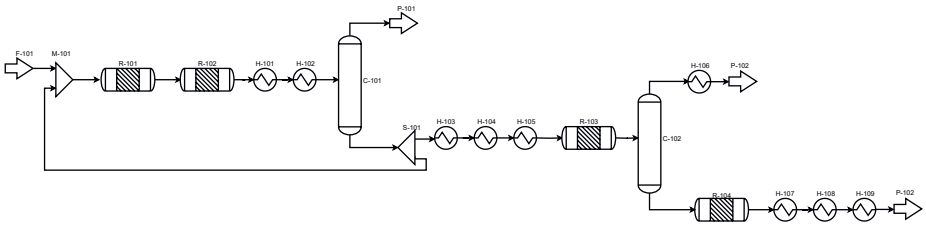


Figure 7.10: An example of process flowsheet diagram of group C1, number of episodes = 250.

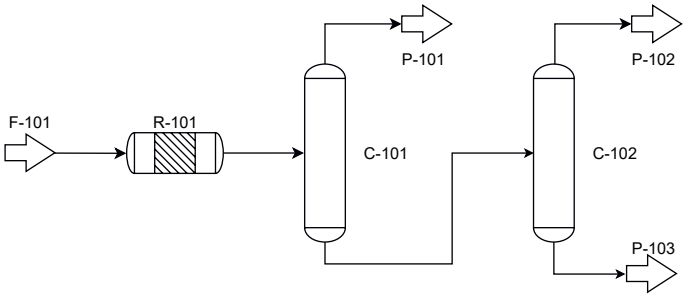


Figure 7.11: An example of process flowsheet diagram of group E3, number of episodes = 1.

with the highest reward (106.84 M€) from the C1 group. The corresponding design and operating variables are detailed in Table 7.4. The feed (F-101) passes through two consecutive PFRs (R-101 and R-102), where products C and D are produced by the esterification of A and B. The resulting quaternary mixture is then separated in a distillation column (C-101). The overhead product from C-101 is further distilled in another column (C-102). In this setup, 90% of the bottom product is recycled and mixed back into the feed, while the mixture of B and D exits through product stream P-104. Additionally, C with 77.4% purity is obtained from product stream P-102, which increases by 37.2% compared with the flowsheet obtained from the shortcut methods, and pure D is obtained from product stream P-103. For the E3 group, the highest-scoring flowsheet achieved 107.53 M€, representing a 0.65% increase compared to the C1 group. As shown in Figure 7.13, the flowsheet generated by the E3 group is similar to that of C1, with one notable difference: the inclusion of an additional cooler (H-101) before the second distillation column (C-102). At lower temperatures, the vapor-liquid equilibrium conditions change, which can favor the separation of components with higher relative volatilities. This modification potentially makes C more readily separable due to its properties at lower temperatures, resulting in a slightly higher purity of C in the E3 flowsheet compared to C1. Both flowsheets utilize two consecutive reactors, likely because the predefined reactor length range (3 to 30 meters) is too short.

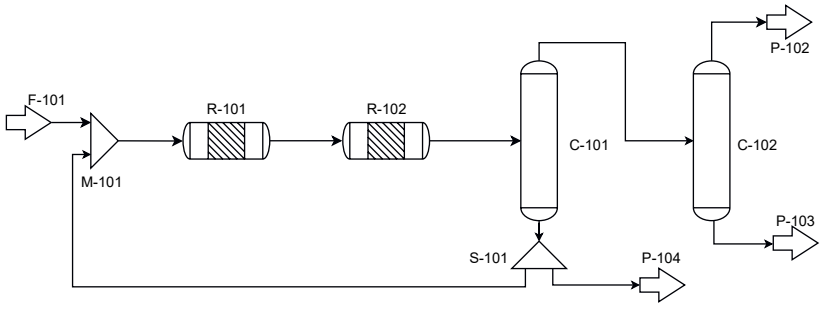


Figure 7.12: The flowsheet with the highest score (106.84 M€) from C1.

Table 7.3: Design variables of unit operations for the flowsheet shown in Figure 7.9

Unit operation	Design variable	Unit	Value for best run
R-101	Length	m	8.07
T-101	D/F ratio	\	0.53
S-101	Split ratio	\	0.34

Table 7.4: Design variables of unit operations for the flowsheet shown in Figure 7.12

Unit operation	Design variable	Unit	Value for best run
R-101	Length	m	23.93
R-102	Length	m	24.36
T-101	D/F ratio	\	0.70
T-102	D/F ratio	\	0.70
S-101	Split ratio	\	0.90

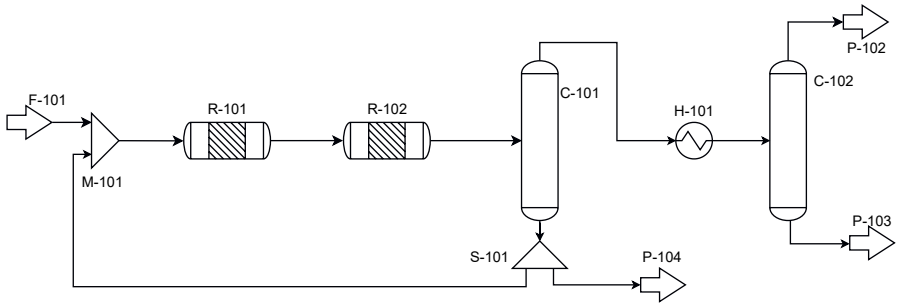


Figure 7.13: The flowsheet with the highest score (107.53 M€) from E3.

Table 7.5: Design variables of unit operations for the flowsheet shown in Figure 7.13

Unit operation	Design variable	Unit	Value for best run
R-101	Length	m	23.65
R-102	Length	m	25.26
C-101	D/F ratio	\	0.69
H-101	Outlet temperature	K	278.15
C-102	D/F ratio	\	0.70
S-101	Split ratio	\	0.90

Further investigation into the flowsheet generated by the E5 group, as shown in Figure 7.14 and detailed in Table 7.6, achieves a highest score of 95.43 M€, which is 10.68% and 11.25% lower than the scores from C1 and E3, respectively. The flowsheet from E5 includes several unsuitable designs: two unnecessary reactors are placed after the distillation columns (C-101 and C-102), and an unnecessary heat exchanger (H-101) is added before the product stream (P-101). The underlying reason is that discrepancies in the reward function and design variables between the shortcut and rigorous simulators can introduce additional bias, leading to lower scores for the highest-scoring flowsheet compared to C1 and E3. E5 requires fewer episodes to reach the highest convergence score threshold (60 M€), as indicated in Table 7.2. This further reflects that E5 has a more stable training process with consistent high-reward flowsheets as the score is the moving average of the previous 100 episodes.

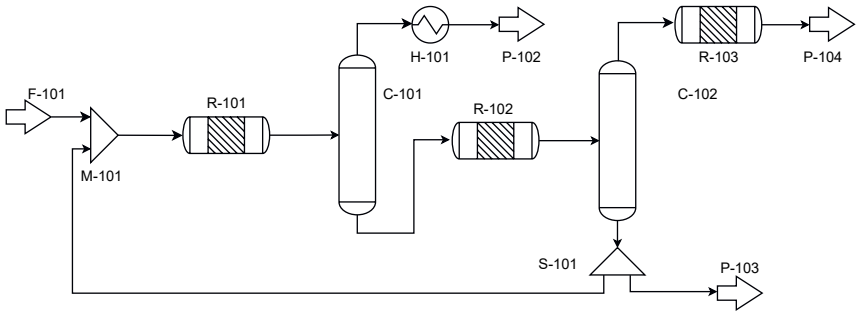


Figure 7.14: The flowsheet with the highest score (95.43 M€) from E5.

7

Table 7.6: Design variables of unit operations for the flowsheet shown in Figure 7.14

Unit operation	Design variable	Unit	Value for best run
R-101	Length	m	27.22
C-101	D/F ratio	\	0.46
H-101	Outlet temperature	K	279.23
R-102	Length	m	23.29
C-102	D/F ratio	\	0.46
R-103	Length	m	13.76
S-101	Split ratio	\	0.9

Figure 7.7 depicts the generated flowsheet with the highest score of 93.09 M€, which is a decrease of 12.87% compared to group C1. One clear unsuitable design is that one reactor (R-102) is included after the distillation column (C-101), which

is quite similar to the flowsheet from E5. Additionally, C with 57.9% purity is obtained from product stream P-102, which is comparatively lower than the other groups. The possible underlying reason is that the learned semantics from the pre-trained encoder greatly mismatched the task in the fine-tuning RL task. This leads to a suboptimal solution within the same training episodes.

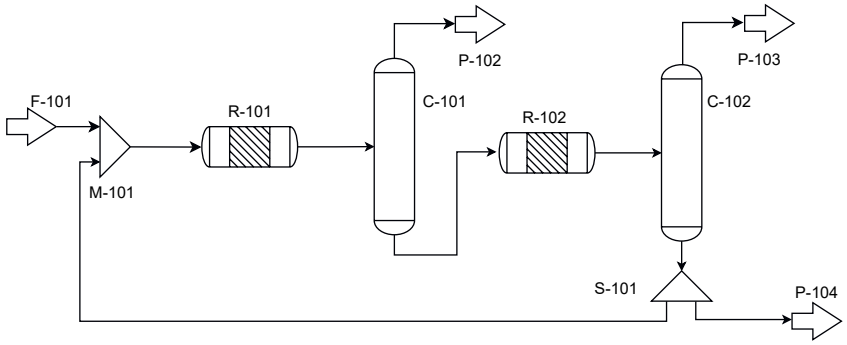


Figure 7.15: The flowsheet with the highest score (93.09 M€) from E6.

Table 7.7: Design variables of unit operations for the flowsheet shown in Figure 7.15

Unit operation	Design variable	Unit	Value for best run
R-101	Length	m	29.24
C-101	D/F ratio	\	0.66
R-102	Length	m	22.40
C-102	D/F ratio	\	0.30
S-101	Split ratio	\	0.83

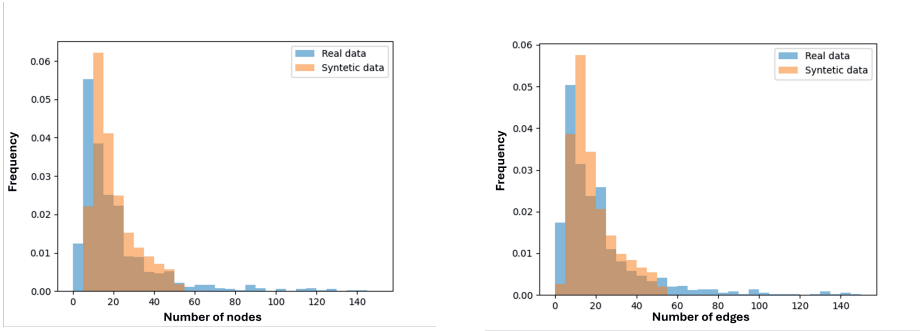
7.4.3. DISCUSSION ON NEGATIVE TRANSFER

Transfer learning not only contributes to higher convergence scores but also reduces training time, as demonstrated by the E3 group. However, in other groups—particularly those employing T2 strategies—negative transfer becomes evident. For T1 strategies, this effect is especially singificant in the E4 and E5 groups, which, unlike E3, also transfer the actor-level 3 and critic networks, respectively. A key factor in the negative transfer observed for the E4 group is the substantial discrepancy between the design/variable ranges used in the pre-training phase and those used in the fine-tuning phase. For instance, while the reactor length range for pre-training spans 0.05 to 20 meters, it shifts to 3 to 30 meters during fine-tuning. Such a mismatch can initially place the model in a

suboptimal portion of the design space, slowing adaptation and ultimately affecting the final convergence score within the same training episodes. Similarly, in the E5 group, the potential cause of negative transfer lies in the differences between the pre-training and fine-tuning objective functions. For example, during fine-tuning, we introduced various utility supplies for the heat exchanger (e.g., low/medium/high-pressure steam) to capture different temperature differences. By contrast, the utility supply was fixed during pre-training. This shift in how the objective function is formulated places the critic networks in a different operating regime. As a result, the networks struggle to accurately approximate the new objective function, ultimately leading to negative transfer.

The training data used for the VAE [38] appears to be the potential cause of negative transfer for the T2 strategy. As shown in Figure 7.16, the VAE was pre-trained on a dataset comprising both real (simulation files and flowsheet data from the literature) and synthetic datasets. For the node distribution, the real dataset averages 21.51 nodes per flowsheet, while the synthetic dataset averages 18.50. In terms of edges, the real dataset averages 23.02 edges per flowsheet, whereas the synthetic dataset averages 18.23. Additionally, Figure 7.16 illustrates that the real dataset has a long right tail with some flowsheets containing up to 200 nodes and edges. Clearly, both the real and synthetic datasets include significantly larger topologies than those used in the fine-tuning phase, where the final flowsheet contains only around 10 nodes. Additionally, the processes in the pre-training dataset differ significantly from the target process. For instance, many of the pre-training examples only involve separation processes, such as benzene-toluene separation or ethanol-water distillation. These are inherently different from the target process in our case study—a combined esterification and separation process—which poses challenges for effective transfer learning. Furthermore, only the flowsheet topology (without edge features) was used when pre-training the VAE, whereas the fine-tuning phase includes edge attributes such as temperature, pressure, and molecular fractions. This discrepancy in both the scale of flowsheet topologies and the nature of edge features potentially places the model in a suboptimal regime at the fine-tuning phase, ultimately contributing to the observed negative transfer.

For future works, several approaches could potentially help mitigate negative transfer in both the pre-training and fine-tuning phases. In pre-training, we suggest using unit operations and objective functions similar to those in the fine-tuning phase. For instance, if the fine-tuning phase involves a compressor or a catalyst-based reactor, one could implement or reuse corresponding shortcut models with comparable variable bounds and the same objective function. If a shortcut model is unavailable, surrogate models can serve as an alternative. A surrogate model is a computationally efficient approximation of a complex system,



(a) Number of unit operations per flowsheet

(b) Number of streams per flowsheet

Figure 7.16: Graph distributions for the synthetic and the real dataset.

enabling rapid predictions, analysis, and optimization without sacrificing critical accuracy. Recent studies have shown the effectiveness of surrogate modeling for processes such as distillation columns [40] and reactors with catalysts [41]. For VAEs, it is crucial that the pre-training dataset closely resembles the fine-tuning dataset, such as the number and types of unit operations, node and edge features, as well as the same target process with different topologies. This alignment can potentially reduce negative transfer. Additionally, supervised pre-training—such as cost prediction for flowsheets or imitation learning—offers another potential strategy. Imitation learning is a machine learning approach in which an agent acquires behaviors or decision-making strategies by mimicking demonstrations provided by an expert. However, collecting a sufficiently large dataset of flowsheets with labels, or obtaining expert demonstrations for imitation learning, remains a significant challenge in practice.

For the fine-tuning phase, our results indicate that both the actor networks for continuous decisions and the critic network have a vital impact on transfer learning performance. If there is a substantial mismatch in design/operating variable ranges or objective functions between the pre-training and fine-tuning phases, we recommend refraining from transferring these two components and instead training them from scratch. Another possible strategy is to initially freeze or remove the actor networks to focus on optimizing the topology first, then apply a separate method—such as a genetic algorithm—to optimize the design/operating variables [20]. For VAEs, if the flowsheet distribution in the pre-training dataset diverges significantly from the target flowsheet distribution in fine-tuning, we likewise advise training the model from scratch. Alternatively, advanced techniques like Low-Rank Adaptation (LoRA) can be leveraged to help manage large parameter spaces and reduce the need for extensive retraining. For instance, applying LoRA to the actor network constrains updates to low-rank factors in the policy

parameters, ensuring that the core policy representation remains largely intact. Similarly, using LoRA on the critic or on an encoder network focuses updates on a limited subset of parameters relevant to value estimation or feature extraction, respectively. In each case, the low-rank constraint could potentially prevent the overwriting of beneficial pre-trained representations, thereby reducing the risk of negative transfer.

In conclusion, the learning curves and optimal flowsheets demonstrate that suitable transfer learning can potentially accelerate the training speed and boost the convergence score. Specifically, within T1 transfer learning strategies, the most suitable transfer learning strategy may involve transferring the encoder and the first two actor levels (E3), corresponding to the ability to accurately interpret states and determine the structure of the discrete process unit network, respectively. Conversely, transferring the design variables and critic network may hinder exploration and lead to suboptimal solutions. Moreover, as transfer learning T2 shows, it is of great importance to align pre-training tasks closely with the target RL tasks to achieve meaningful performance improvements.

7.5. CONCLUSIONS

We propose utilizing transfer learning to enhance the learning process of RL in process design. Specifically, we investigate two transfer learning strategies: (i) transfer learning from shortcut process simulators to rigorous simulators (T1) and (ii) transfer learning from process VAE (T2). Our results demonstrate that appropriate transfer learning can significantly improve both learning efficiency and convergence scores. For instance, transferring pre-trained encoder and actor levels 1 and 2 (E3) reduced convergence training episodes by 63.88% and increased convergence scores by 0.65% compared to the group without transfer learning (C1).

Conversely, transfer learning can also negatively impact the learning process. Transferring continuous action networks and critic networks can hinder agent learning if there are substantial discrepancies in decision range and reward function between pre-trained and fine-tuning environments. Additionally, pre-training the encoder with existing flowsheet data yielded only minor improvements in convergence scores and sometimes slowed down the fine-tuning process. This suggests that pre-trained process data should match the complexity of the fine-tuning task.

This work provides guidance on effectively utilizing transfer learning strategies to enhance the efficiency and effectiveness of RL in process design. Future research could explore incorporating more comprehensive flowsheet topology data with additional unit operations and stream information to pre-train the graph encoder. Additionally, RL could potentially integrate with other GenAI tools

to support further engineering workflow [39].

7.6. ACKNOWLEDGEMENTS

This work was supported by the Netherlands Organisation for Scientific Research (NWO) under the Veni grant.

REFERENCES

[1] A. Mitsos, N. Asprion, C. A. Floudas, M. Bortz, M. Baldea, D. Bonvin, A. Caspari, and P. Schäfer, *Challenges in process optimization for new feedstocks and energy sources,* Computers & Chemical Engineering **113**, 209 (2018).

[2] J. J. Siirola and D. F. Rudd, *Computer-aided synthesis of chemical process designs. from reaction path data to the process task network,* Industrial & Engineering Chemistry Fundamentals **10**, 353 (1971).

[3] V. Mahalec and R. Motard, *Procedures for the initial design of chemical processing systems,* Computers & Chemical Engineering **1**, 57 (1977).

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, *Playing atari with deep reinforcement learning,* (2013), 10.48550/ARXIV.1312.5602, arXiv:1312.5602 [cs.LG] .

[5] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, *A general reinforcement learning algorithm that masters chess, shogi, and go through self-play,* Science **362**, 1140 (2018).

[6] R. Nian, J. Liu, and B. Huang, *A review on reinforcement learning: Introduction and applications in industrial process control,* Computers & Chemical Engineering **139**, 106886 (2020).

[7] M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen, *Molecular de-novo design through deep reinforcement learning,* Journal of Cheminformatics **9** (2017), 10.1186/s13321-017-0235-x.

[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning An Introduction* (A Bradford Book) p. 552.

[9] Q. Gao and A. M. Schweidtmann, *Deep reinforcement learning for process design: Review and perspective,* Current Opinion in Chemical Engineering **44**, 101012 (2024).

[10] A. Perera, P. Wickramasinghe, V. M. Nik, and J.-L. Scartezzini, *Introducing reinforcement learning to the energy system design process,* Applied Energy **262**, 114580 (2020).

[11] A. Khan and A. Lapkin, *Searching for optimal process routes: A reinforcement learning approach,* Computers & Chemical Engineering **141**, 107027 (2020).

[12] L. I. Midgley, *Deep reinforcement learning for process synthesis,* (2020), 10.48550/ARXIV.2009.13265, arXiv:2009.13265 [cs.LG] .

[13] Q. Göttl, Y. Tönges, D. G. Grimm, and J. Burger, *Automated flowsheet synthesis using hierarchical reinforcement learning: Proof of concept,* Chemie Ingenieur Technik **93**, 2010 (2021).

[14] A. A. Khan and A. A. Lapkin, *Designing the process designer: Hierarchical reinforcement learning for optimisation-based process design,* Chemical Engineering and Processing - Process Intensification **180**, 108885 (2022).

[15] S. J. Plathottam, B. Richey, G. Curry, J. Cresko, and C. O. Iloeje, *Solvent extraction process design using deep reinforcement learning,* Journal of Advanced Manufacturing and Processing **3** (2021), 10.1002/amp2.10079.

[16] L. Stops, R. Leenhouts, Q. Gao, and A. M. Schweidtmann, *Flowsheet generation through hierarchical reinforcement learning and graph neural networks,* AIChE Journal **69** (2022), 10.1002/aic.17938.

[17] S. Reynoso-Donzelli and L. A. Ricardez-Sandoval, *Enhancing process flowsheet design through masked hybrid proximal policy optimization,* IFAC-PapersOnLine **58**, 646 (2024).

[18] S. Reynoso-Donzelli and L. A. Ricardez-Sandoval, *A reinforcement learning approach with masked agents for chemical process flowsheet design,* AIChE Journal **71** (2024), 10.1002/aic.18584.

[19] H. Tan, X. Hong, Z. Liao, J. Sun, Y. Yang, J. Wang, and Y. Yang, *Combining reinforcement learning with mathematical programming: An approach for optimal design of heat exchanger networks,* Chinese Journal of Chemical Engineering **69**, 63 (2024).

[20] Q. Göttl, H. Asif, A. Mattick, R. Marzilger, and A. Plinge, *Automated design in hybrid action spaces by reinforcement learning and differential evolution,* in *KI 2024: Advances in Artificial Intelligence* (Springer Nature Switzerland, 2024) pp. 292–299.

[21] S. C. P. A. van Kalmthout, L. I. Midgley, and M. B. Franke, *Synthesis of separation processes with reinforcement learning,* (2022), 10.48550/ARXIV.2211.04327, arXiv:2211.04327 [cs.LG] .

[22] Q. Göttl, J. Pirnay, J. Burger, and D. G. Grimm, *Deep reinforcement learning enables conceptual design of processes for separating azeotropic mixtures without prior knowledge,* Computers & Chemical Engineering **194**, 108975 (2025).

[23] M. Hussain, J. J. Bird, and D. R. Faria, *A study on cnn transfer learning for image classification,* in *Advances in Computational Intelligence Systems* (Springer International Publishing, 2018) pp. 191–202.

[24] S. Ruder, M. E. Peters, S. Swayamdipta, and T. Wolf, *Transfer learning in natural language processing,* in *Proceedings of the 2019 Conference of the North* (Association for Computational Linguistics, 2019).

[25] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, *A comprehensive survey on transfer learning,* Proceedings of the IEEE **109**, 43 (2021).

[26] D. Wang, J. Bao, M. Zamarripa-Perez, B. Paul, Y. Chen, P. Gao, T. Ma, A. Noring, A. Iyengar, D. Schwartz, E. Eggleton, Q. He, A. Liu, O. Marina, B. Koeppel, and Z. Xu, *Reinforcement learning for automated conceptual design of advanced energy and chemical systems,* (2022), 10.21203/rs.3.rs-2248780/v1.

[27] Y. Tian, A. Akintola, Y. Jiang, D. Wang, J. Bao, M. A. Zamarripa, B. Paul, Y. Chen, P. Gao, A. Noring, A. Iyengar, A. Liu, O. Marina, B. Koeppel, and Z. Xu, *Reinforcement learning-driven process design: A hydrodealkylation example,* Systems and Control Transactions **3**, 387 (2024).

[28] Q. Gao, H. Yang, S. M. Shanbhag, and A. M. Schweidtmann, *Transfer learning for process design with reinforcement learning,* in *33rd European Symposium on Computer Aided Process Engineering* (Elsevier, 2023) pp. 2005–2010.

[29] W. Zhang, L. Deng, L. Zhang, and D. Wu, *A survey on negative transfer,* IEEE/CAA Journal of Automatica Sinica **10**, 305 (2023).

[30] A. M. Schweidtmann, J. G. Rittig, A. König, M. Grohe, A. Mitsos, and M. Dahmen, *Graph neural networks for prediction of fuel ignition quality,* Energy & Fuels **34**, 11395 (2020).

[31] Z. P. Xu and K. T. Chuang, *Kinetics of acetic acid esterification over ion exchange catalysts,* The Canadian Journal of Chemical Engineering **74**, 493 (1996).

[32] J. Achiam, D. Held, A. Tamar, and P. Abbeel, *Constrained policy optimization,* (2017), 10.48550/ARXIV.1705.10528, arXiv:1705.10528 [cs.LG] .

[33] J. García and F. Fernández, *A comprehensive survey on safe reinforcement learning,* Journal of Machine Learning Research **16**, 1437 (2015).

[34] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms,* (2017), 10.48550/ARXIV.1707.06347, arXiv:1707.06347 [cs.LG] .

[35] G. Vogel, L. Schulze Balhorn, and A. M. Schweidtmann, *Learning from flowsheets: A generative transformer model for autocompletion of flowsheets,* Computers & Chemical Engineering **171**, 108162 (2023).

[36] E. Hirtreiter, L. Schulze Balhorn, and A. M. Schweidtmann, *Toward automatic generation of control structures for process flow diagrams with large language models,* AIChE Journal **70** (2023), 10.1002/aic.18259.

[37] L. S. Balhorn, Q. Gao, D. Goldstein, and A. M. Schweidtmann, *Flowsheet recognition using deep convolutional neural networks,* in *14th International Symposium on Process Systems Engineering* (Elsevier, 2022) pp. 1567–1572.

[38] M. F. Theisen, Q. Gao, and A. M. Schweidtmann, *Representation learning for process topologies using variational autoencoders,* (2024), in preparation.

[39] A. M. Schweidtmann, *Generative artificial intelligence in chemical engineering,* Nature Chemical Engineering **1**, 193 (2024).

[40] J. Lu, Q. Wang, Z. Zhang, J. Tang, M. Cui, X. Chen, Q. Liu, Z. Fei, and X. Qiao, *Surrogate modeling-based multi-objective optimization for the integrated distillation processes,* Chemical Engineering and Processing - Process Intensification **159**, 108224 (2021).

[41] G. Lastrucci, M. F. Theisen, and A. M. Schweidtmann, *Physics-informed neural networks and time-series transformer for modeling of chemical reactors,* in *34th European Symposium on Computer Aided Process Engineering / 15th International Symposium on Process Systems Engineering* (Elsevier, 2024) pp. 571–576.

APPENDIX

Table 7.8: Hyperparameters for the architecture and pre-training procedure of the actor-critic agent.

Parameter		Value
Learning rate	α	0.0002
Policy clipping factor	ϵ	0.3
Discount factor	γ	1.0
λ-return factor	λ	0.95
Number of steps after which the agent learns	n_{B}	60
Batch size during learning	n_{MB}	30
Number of epochs	n_{E}	4
Weight for loss of level 1 actor	c_0	0.1
Weight for loss of level 2 actor	c_1	1.0
Weight for loss of level 3 actor	c_2	0.5
Weight for loss of critic	c_3	0.2
Weight for entropy of level 1 actor	d_1	0.001
Weight for entropy of level 2 actor	d_2	0.3
Weight for entropy of level 3 actor	d_3	0.001
Hidden layers edge processing for fingerprint	-	10
Message passing steps for fingerprint	-	6
Hidden layers dimension level 1 actor	-	12
Hidden layers dimension level 2 actor	-	256
Hidden layers dimension level 3 actor	-	256
Hidden layers dimension critic	-	256
Feature size flowsheet fingerprint	-	50

Table 7.9: Hyperparameters for the architecture and fine-tuning procedure of the actor-critic agent with rigorous simulator.

Parameter		Value
Actor level 1 learning rate	α_1	1×10^{-6}
Actor level 2 learning rate	α_2	1×10^{-5}
Actor level 3 learning rate	α_3	1×10^{-6}
Critics network learning rate	α_4	1×10^{-6}
Policy clipping factor	ϵ	0.3
Discount factor	γ	1.0
λ-return factor	λ	0.95
Number of steps after which the agent learns	n_{B}	20
Batch size during learning	n_{MB}	30
Number of epochs	n_{E}	4
Weight for loss of level 1 actor	c_0	0.1
Weight for loss of level 2 actor	c_1	1.0
Weight for loss of level 3 actor	c_2	0.5
Weight for loss of critic	c_3	0.2
Weight for entropy of level 1 actor	d_1	0.001
Weight for entropy of level 2 actor	d_2	0.3
Weight for entropy of level 3 actor	d_3	0.001
Hidden layers edge processing for fingerprint	-	10
Message passing steps for fingerprint	-	6
Hidden layers dimension level 1 actor	-	12
Hidden layers dimension level 2 actor	-	256
Hidden layers dimension level 3 actor	-	256
Hidden layers dimension critic	-	256
Feature size flowsheet fingerprint	-	50

V

INTEGRATION OF MOLECULAR AND PROCESS DESIGN

8

MULTI-AGENT SYSTEMS FOR CHEMICAL ENGINEERING: A REVIEW AND PERSPECTIVE

Large language model (LLM)-based multi-agent systems (MASs) are a recent but rapidly evolving technology with the potential to transform chemical engineering by decomposing complex workflows into teams of collaborative agents with specialized knowledge and tools. This review surveys the state-of-the-art of MASs within chemical engineering. While early studies demonstrate promising results, scientific challenges remain, including the design of tailored architectures, integration of heterogeneous data modalities, development of foundation models with domain-specific modalities, and strategies for ensuring transparency, safety, and environmental impact. As a young but fast-moving field, MASs offer exciting opportunities to rethink chemical engineering workflows.

This chapter has been published as "Multi-agent systems for chemical engineering: A review and perspective" by S. Rupprecht, Q. Gao, T. Karia, & A. M. Schweidtmann, Current Opinion in Chemical Engineering,51, 101209, (2026)

Author contribution. This work was led by S. Rupprecht as first author. The contribution of the author of this dissertation consisted of writing the introduction and revising and reviewing the entire manuscript. The contribution claimed in this dissertation is limited to these parts; the chapter is included as a perspective that motivates the orchestration layer discussed in this thesis.

8.1. INTRODUCTION

The advent of large language models (LLMs), such as GPT-4 [1], has marked a technological leap, enabling advanced capabilities in language translation, multimodal data processing, mathematical reasoning, and code generation [2]. These advances have facilitated progress in various domains, including chemistry and materials science. Likewise, there is growing interest in developing and applying LLMs in the chemical engineering domain [3, 4] and more specifically to process systems engineering [5–7].

Although LLMs have demonstrated strong capabilities in various domains, their application in chemical engineering remains limited due to multiple fundamental shortcomings. Chemical engineering workflows are inherently complex, spanning multiple scales and relying heavily on specialized simulation tools and external data sources. Such workflows cannot be reliably replicated or replaced by standalone LLMs yet. Trained as end-to-end black box models, LLMs are prone to hallucinations arising from biases and gaps in their training data, producing confident yet incorrect outputs. Employing such standalone LLMs lacks transparency and interpretability, and can even lead to significant safety risks. Critically, general-purpose LLMs do not possess an intrinsic understanding of physical laws, such as thermodynamic constraints or mass and energy balances, frequently resulting in outputs that are physically invalid [8, 9]. Thereby, LLMs often fail to solve domain-specific, multi-step problems that demand physical reasoning and a system-level perspective. Without access to relevant tools and trusted data sources, even the most advanced LLMs remain fundamentally constrained in their capabilities.

LLM-based multi-agent systems (MASs) offer a promising methodology to address these challenges. An agent is most generally defined as an entity that perceives its environment and independently executes actions upon the environment on behalf of its owner [10, 11]. In contrast to rule-based agentic frameworks, an LLM-based agent is a dynamic computational entity that is capable of communicating through natural language and can attempt to reason over a broader range of tasks [12]. In the context of MASs, LLM-based agents address the shortcomings of standalone LLMs by extending their capabilities to interact with tools and databases, enabling them to access external knowledge for more informed decision-making and increasing the transparency and traceability of information. MASs elicit inter-agent cooperation, coordination, and negotiation to accomplish individual or shared goals [11]. MASs subdivide complex tasks into smaller subtasks and assign roles to distinct agents that can interact with each other [13, 14]. Therein, single agents are usually LLMs that are equipped with carefully selected external knowledge in the form of executable tools, access to databases, and literature to perform specific tasks [2]. The individual agents need to possess

a certain level of autonomy, decision-making capabilities, proactiveness, and communication skills to make the MAS performative in dynamic and distributed environments.

MASs have already shown success in other domains [15, 16]. Notable examples of MASs are in chemistry [17–19], drug discovery [20], material design [21], scientific discovery [22–24], medicine [25] and code generation [26]. Recent production-ready frameworks such as Clint and CodeRabbit further showcase the use of multi-agent collaboration in software engineering tasks, including autonomous development, code review, and project management. Recently, several review and perspective articles have specifically focused on the potential of MASs in subfields of chemical engineering [9, 27, 28]. However, a comprehensive MAS for chemical engineering has not been proposed or developed yet.

In this article, we review the developments of MASs within the overarching chemical engineering domain and provide an outlook for further development and applications. In Section 8.2, we review the recent studies applying tailored, LLM-supported MASs demonstrating the potential of MASs for chemical engineering applications. Furthermore, in Section 8.3, we discuss the fundamental challenges that must be addressed to enable cross-scale integration in chemical engineering workflows with MASs. Finally, we conclude the paper in Section 8.4.

8.2. STATE OF THE ART

In this section, we first introduce the core concepts of MASs (Section 8.2.1) and then review literature on applications of MASs in chemical engineering (Section 8.2.2).

8.2.1. MULTI-AGENT SYSTEMS

MASs are distributed systems composed of multiple agents that interact to achieve individual or collective goals [2]. While traditional supervised models learn static mappings, agents can dynamically reason over tasks, invoking tools or querying data as needed. These capabilities facilitate iterative and reflective interaction in agents, thereby improving transparency and adaptability compared to standalone LLMs. Human-in-the-loop guidance is required to ensure the alignment of the MAS conversation with the overall objective of the human user. A human-in-the-loop should also be in place to address shortcomings of agents' outputs during conversation. Hence, MASs with human-in-the-loop provide a natural framework for streamlining chemical engineering workflows that require complex decision-making. While MASs were proposed back in the 1990s for autonomous agents and distributed AI [11], they have received significant attention recently due to the advent of LLMs.

For modern MASs, agents are typically LLMs capable of interacting with tools

by selecting suitable tools, providing tools' input arguments, and interpreting tool outputs. Agents are also capable of querying structured information from relational databases (e.g., via SQL [29]) and knowledge graphs (e.g., via Cypher [30]), enabling access to structured domain knowledge. A knowledge graph is a structured network of entities (nodes) and their relationships (edges) that encodes meaning and context, enabling both humans and machines to query, integrate, and reason over connected data. For unstructured information—such as scientific literature, reports, or manuals—retrieval-augmented generation (RAG) [31] techniques are used to extract relevant content. Large unstructured text corpora are segmented into smaller text chunks during preprocessing. Each chunk is then transformed into a semantically rich vector representation, or embedding, and stored in a vector database. RAG finally allows the retrieval of semantically similar text snippets during inference, given a user query. Recent extensions adapt RAG for graph-based retrieval, i.e., Graph RAG, [32–34] by transforming unstructured text into knowledge graphs. The knowledge graphs used for Graph RAG-based approaches offer the advantage of capturing top-level relations between pieces of information, compared to isolated text snippets in the case of standard RAG methods [32].

To facilitate communication among various agents to develop MASs, several open-source libraries such as AutoGen [13], crewAI [35] by Microsoft, and LangGraph [36] by LangChain have been proposed. There exist many more open-source MAS frameworks, such as CAMEL [37] and MetaGPT [38]. Such libraries offer toolkits to design MASs and infrastructure to use a variety of LLMs as drop-in models within agents. For instance, LangChain supports interfaces for integrating LLMs from multiple providers, including OpenAI, Ollama, and Anthropic. MASs, of course, cannot eliminate the hallucinations of the LLMs they use within agents. For instance, [39] explored a LangChain prebuilt chatbot for pharmaceutical engineering and observed significant propagation of hallucinations in the case of knowledge graph information retrieval.

LangGraph [36] is widely used for scientific applications and supports flexible conversation via graph structures. In LangGraph's graph structure, nodes represent agents, and edges represent communication. LangGraph [36] is also capable of differentiating between unconditional and conditional edges. Unconditional edges imply that there is a predetermined next agent, independent of the contribution of the previous agent. Conditional edges indicate that the next agent is dynamically inferred during inference, depending on the state of the conversation. These frameworks can incorporate human agents, enabling expert oversight and intervention, enhancing reliability, interpretability, and control - crucial for safety-critical domains like chemical engineering.

Figure 8.1 illustrates five basic architecture types of MASs [15, 16, 36, 40]: (I)

Chain — a linear sequence of agents using unconditional edges only; (II) Directed acyclic graph (DAG) — incorporates dynamic edges to enable branching paths; (III) Directed cyclic graph (DCG) — allows iterative workflows by including cycles in the graph; (IVa) Centralized — a supervisor agent coordinates specialized sub-agents; (IVb) Nested hierarchy — an extension of the centralized model where the supervisor manages modular sub-systems, each with its own internal structure; (V) Network — fully connected agents that communicate freely and can independently terminate or respond to tasks. The architectures I to V are organized with increasing inter-agent connectivity, which can increase the flexibility of the MAS's behaviour. Such increased flexibility could allow for more feedback mechanisms between agents and humans-in-the-loop, and can potentially lead to an increased level of autonomy of the MAS. The choice of architecture depends on the nature of the application, including task complexity, communication needs, and coordination overhead. Each architecture offers trade-offs in terms of scalability, interpretability, and adaptability.

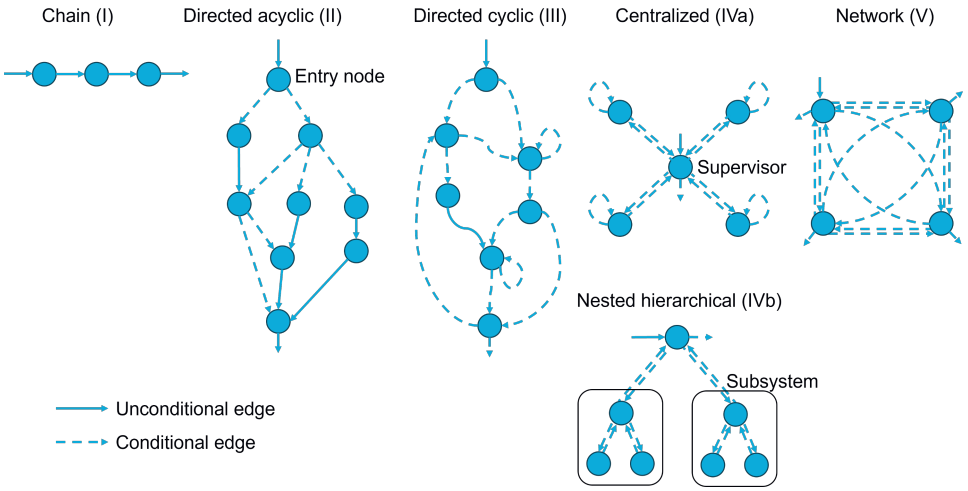


Figure 8.1: Architecture options for multi-agent systems. Multi-agent systems can be modeled as graphs [36] with agents as nodes and edges as conversation routes. Unconditional edges imply that there is a predetermined next agent, independent of the contribution of the previous agent. Conditional edges indicate that the next agent is dynamically inferred during inference, depending on the state of the conversation. The chain architecture (I) is a fixed sequence of agents connected with conditional edges. A directed acyclic conversation graph (II) contains conditional edges. A directed cyclic conversation graph (III) includes loops where substeps of the conversation can be repeated. In a centralized architecture (IVa), there is a single supervisor agent managing multiple subordinate agents. A nested hierarchical structure (IVb) includes a supervisor managing multiple sub-systems, which can consist of single agents or independent, customized architectures. The network architecture (V) puts each agent in direct contact with every other agent independently.

8.2.2. MULTI-AGENT SYSTEMS IN CHEMICAL ENGINEERING

In Table 8.1, we summarize recent studies showcasing the application of MASs for chemical engineering tasks [39, 41–49]. [41] propose a MAS that suggests process design improvements based on existing PFDs. [45] use a MAS to balance economic and environmental objectives in gas–oil separation. MASs have also been employed for general process problem-solving [44], autonomous industrial control [42], and generating PFDs or P&IDs from natural language descriptions [43, 46]. Others use agents to identify operational constraints in process optimization [47] or to build structured knowledge graphs from multimodal data for schematic synthesis via GraphRAG [48]. Approaches to MAS applications in quantum chemistry [49] and pharmaceutical engineering [39] are included in the review as well. Now, we examine the publications along five aspects: architecture of the MASs, tool and database integration, modalities of data used, and human integration.

Table 8.1: Overview of multi-agent systems (MASs) in the chemical engineering domain. The existing literature is categorized by architecture type, task addressed, examples of tool and database integration, human integration, and framework used for implementation. DCG refers to the directed cyclic graph architecture type. RAG refers to retrieval augmented generation.

Literature	Task	MAS architecture	Tools	Database integration	Human integration	Framework
[41]	Process improvement	Chain	ChemicalAid, Literature data extraction	-	-	OpenAI
[42]	Industrial control	DCG	Arduino temperature control lab	-	-	CrewAI
[43]	P&ID generation from text	DCG	MS Visio	-	✓	n/a
[44]	General operations assistance	Centralized	Stack Overflow, Wolfram Alpha	Graph RAG	-	n/a
[45]	Operational decision-making	DCG	Pyomo, Aspen HYSYS	RAG	-	LangGraph
[46]	PFD and P&ID generation	DCG	-	RAG& Graph RAG	✓	n/a
[47]	Process optimization	DCG	IDAES simulation	-	-	AutoGen
[48]	PFD and P&ID generation	DCG	Web search, Multimodal information extraction	Graph RAG	✓	n/a
[49]	Computational quantum chemistry task support	Nested hierarchical	DFT calculations RDKit, ORCA, xTB	-	✓	LangGraph & LangChain
[39]	Pharameutical engineering chatbot	Chain	-	Knowledge graph	-	LangChain

Architecture The DCG (III) is the most common architecture type of MASs in chemical engineering [42, 43, 45–47] (Table 8.1). MASs using a DCG architecture typically mirror the structure of the engineering workflows they support. Feedback

loops and iterative refinement steps within such engineering workflows thus make DCGs the frontrunner architectural choice. We hereafter briefly detail how publications using a DCG MAS architecture facilitate alignment with the corresponding engineering workflow. Afterwards, we detail how previous work addresses tasks using a DCG that does not follow existing engineering workflows. Finally, we inspect how other architecture types, I, IVa, and IVb, are applied in previous work.

[42] suggest mirroring the control and monitoring structure of processes at an industrial scale. This results in a DCG due to the nested feedback loops of common control structures. The DCG presented by [43] aims at the automatic creation of P&IDs in an iterative, cyclic workflow. The user describes individual subsystems of the envisioned P&ID in natural language, which the MAS then translates into a standardized, XML-based representation format each. The representation of each P&ID subsystem is validated by a human-in-the-loop. This sequence of steps continues until all subsystems are added to the P&ID. [45] propose a MAS for a decision-making process for operational setpoint selection of a gas-oil separation plant. The MAS architecture follows three steps: deriving a Pareto front of potential setpoints, selecting a suitable setpoint - considering both economic and environmental aspects, and finally, running a simulation of the plant with the selected setpoint. The decision-making process within the second step closely resembles an iterative human-centric workflow, thus resulting in a DCG MAS architecture: Two agents enter a multi-turn debate, each supporting a different objective: minimal costs or minimal emissions. A third agent thereafter selects a setpoint based on the arguments presented during the debate. [48] employ a DCG architecture to systematically extract and aggregate chemical process information from a plethora of multimodal sources. A meta-agent orchestrates sub-agents that specialize in a certain modality each: images, journal articles, patents, HTML etc. Each agent describes the information obtained from each modality in language. The sum of information is aggregated and assessed, introducing an iterative feedback loop.

Previous work also employs DCG architectures for workflows distinct from human-centric procedures [46, 47]. [46] implement an iterative two-agent MAS that generates natural language descriptions of PFDs and P&IDs. A coordinating agent guides a subordinate agent to retrieve relevant data from curated databases. The resulting response is evaluated iteratively by a critique agent that provides feedback using an LLM-as-a-judge mechanism or human feedback until a generated response satisfies the critique agent's judgement. [47] define process operating constraints using a DCG that differs from heuristics-based human-centric workflows. Given a user's description of a process, an initial agent defines bounds for the process's decision variables. Next, a group of agents derives suitable pa-

rameter values in a guided, iterative collaborative workflow.

The chain (I) and DAG (II) architectures enforce strictly linear or branching workflows without feedback loops. For instance, [41] use a six-agent chain (I) architecture following a predefined sequence of tasks to improve existing PFDs. [39] employ two LLM-backed agents in sequence for entity extraction and response generation, respectively, as a chatbot for pharmaceutical engineering questions. However, the rigid design types (I-II) limit the system's ability to handle iterative refinement or adapt to edge cases, which may constrain performance on more complex or ill-defined tasks. In contrast, DCGs (III) offer a balance between rigidity, which enforces task structure, and flexibility, which allows iteration and adaptation to edge cases.

The centralized (IVa), the nested hierarchical (IVb), and the network architecture (V) favor flexibility over structured flow and align with tasks that lack a predefined engineering sequence. The centralized architecture (IVa) used by [44] assigns decision-making to a central coordinator that delegates tasks to specialized agents. [49] implement the nested hierarchical (IVb) architecture where a top-level agent delegates tasks to specialized subgroups of agents. This publication presents a hybrid form because elements of a network (V) are present: Some subgroup agents interact with agents at different hierarchy levels or agents of other subgroups.

Tool integration A majority of MAS applications in chemical engineering rely on tool integration [41–47, 49]. This highlights the central role of outsourcing tasks to specialized tools from purely LLM-based workflows. Tool integration is thus a crucial enabler in establishing MAS functionality within chemical engineering.

Previous work chooses different ways to interface with the tools listed in Table 8.1, depending on the public availability of tool interfaces. Existing Application Programming Interfaces (APIs) are leveraged by previous work to integrate desired tools into chemical engineering MASs, e.g., Stack Overflow's and Wolfram Alpha's APIs [44], or Microsoft Visio's C# API [43]. If offline software does not provide publicly available interfaces, users must develop custom interfaces to facilitate tool integration. For example, [45] and [47] interact with simulation environments through tailored interfaces to Aspen HYSYS and Pyomo [45], and IDAES [47], respectively.

The distribution of tools across agents is a critical decision that is addressed differently in the reviewed literature. In single-agent systems, all tools are assigned to a single agent, e.g., in the case of ChemCrow [19]. ChemCrow's LLM-powered agent can access 18 tools, requiring it to both select the appropriate tool and formulate the corresponding tool-call arguments to accomplish tasks spanning from automatic synthesis to materials design. In most reviewed MAS examples,

tools are distributed across agents by expertise. For example, El Agente [49] groups agents into modules by expertise. A subagent may have access to a selection of tools that match the domain of expertise, or no tool access. An individual tool may be called by different agents if the tool suits the agents' scope of action, e.g., the execution of Python code in El Agente's architecture[49].

Database integration Across the reviewed literature, external information from databases is retrieved to enhance the domain relevance of LLM-backed agents' prompts. The works differ in how information is retrieved from the external databases. The most common retrieval methods are RAG [45] and Graph RAG [46, 48]. Such RAG-based approaches aim at retrieving information from the external vector database that is semantically similar to a provided query. [45] extract relevant pieces of text from larger reports, e.g., the COP28. Graph RAG database retrieval is applied for advanced code generation [44] and retrieval of process flow and instrumentation descriptions [46, 48]. A different approach to database integration is presented by [39], where raw information is queried from a graph database.

We observe differences in the generation of the databases used in the reviewed literature. The chemical engineering domain is not a classical big data domain [50] as there is a shortage of publicly available relational and graph databases compared to other domains, such as chemistry. Hence, previous work presents approaches to generate tailored structured databases. The source of origin for all databases used by the reviewed literature is unstructured literature [39, 44–46, 48, 51]. Most of these works [39, 44, 46, 48, 51] construct a knowledge graph of text embeddings of the information acquired. Before an extensive knowledge graph can be built, extensive amounts of multimodal information are aggregated first [44, 46, 48, 51]. The modalities include text in combination with equations [51], code [44], tables [44, 51], and figures [44, 48, 51]. To generate text embeddings of all modalities, previous literature commonly employs multimodal LLMs to produce descriptions of images, tables, etc. [44, 48, 51]. The ontology of the underlying knowledge graph is often extracted dynamically by an LLM [44, 48]. An exception to this is the work of [39], who base their knowledge graph construction process on unified, domain-specific ontologies.

Modalities Chemical engineering spans heterogeneous data types: sensor signals, molecular representations (e.g., SMILES, graphs), process flowsheets, P&IDs, and simulation files [52]. Textual information (natural language instructions or additional contextual information) is often inserted directly into LLM prompts. Modalities other than text are typically preprocessed before they are introduced into the MAS or individual LLMs. In the previous section on database integra-

tion, we outlined that previous work aggregates information stored in different modalities in academic literature into a joint knowledge graph of text embeddings [44, 46, 48, 51]. Tables, figures, and code are summarized using multimodal LLMs [44, 46, 48, 51]. Numerical values - such as simulation results [45, 47] or sensor data [42] - are typically embedded into prompts. Chemical processes are represented via natural language descriptions in the work of [46]. Chemical processes can also be represented as graphs, which are used either as an alternative to text (e.g., D-SFILES [41]) or as structured XML inputs [43, 53].

Human integration Most MASs in chemical engineering reviewed so far operate autonomously, and they do not integrate human oversight [41, 42, 44, 45, 47]. Two studies embed human interaction into the MAS architecture [43, 46]: [43] employ a human assessor to validate new additions to a P&ID visually through MS Visio. [46] use a human-in-the-loop to provide feedback on suggestions made during the solution process of the MAS.

Most works reviewed so far focus on solving a specific task in chemical engineering. A notable exception is the work of [44], which addresses general question answering in chemical engineering. In the next section, we outline our perspective on developing a general MAS for chemical engineering.

8.3. PERSPECTIVE

We envision MASs to become an important tool for chemical engineering (Section 8.3.1). After proposing our vision, we discuss integral parts to achieve this vision, including MAS architecture (Section 8.3.2), tool (Section 8.3.3) and database integration (Section 8.3.4), knowledge base integration (Section 8.3.5), and chemical engineering foundation models with domain-specific modalities (Section 8.3.6). In Section 8.3.7, we critically reflect on challenges regarding the transparency and reliability of such systems.

8

8.3.1. OUR VISION

We envision MASs in chemical engineering as interconnected, human-centric collaborators that integrate across different scales, with access to domain-specific tools, databases, and modalities to drive intelligent and transparent decision-making.

Chemical engineering challenges span multiple scales from molecular to plant-wide operations and global supply chains. As illustrated in Figure 8.2, MASs offer a natural way to bridge these layers, with agents specializing in specific tasks while coordinating toward shared objectives. This architecture aligns closely with how chemical engineering work is already organized: in teams of experts

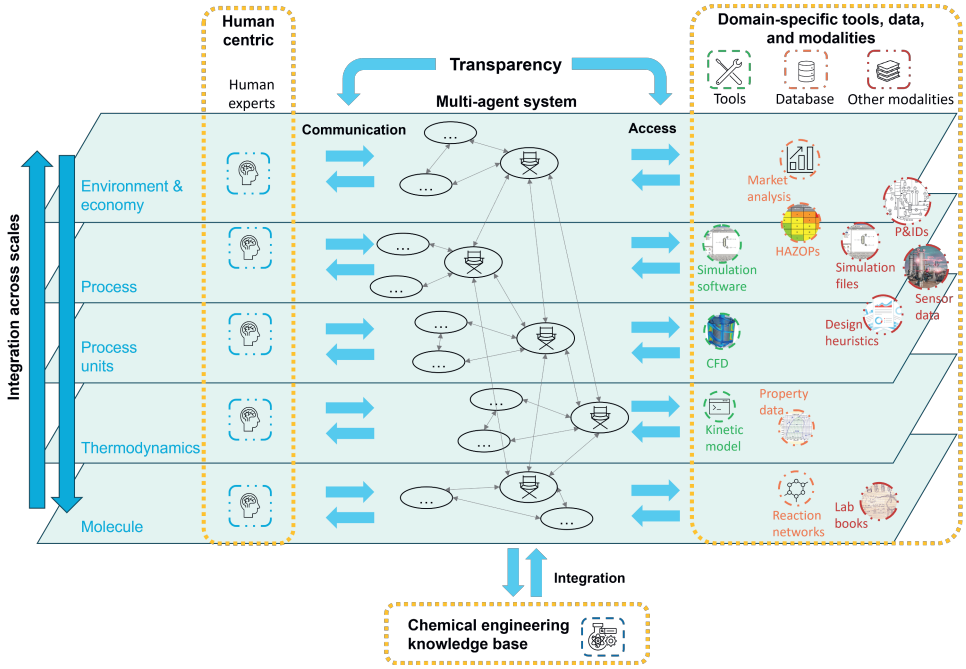


Figure 8.2: Multi-agent systems in chemical engineering as interconnected, human-centric collaborators that integrate across different scales, with access to domain-specific tools, databases, and modalities to drive intelligent and transparent decision-making.

using specialized tools and information. For example, agents equipped with thermodynamic and kinetic models can explore reaction pathways and materials, accelerating discovery at the molecular level. At the process level, operational agents dynamically run optimization routines, balancing yield, energy efficiency, and safety. Across plant and supply chain scales, agents negotiate schedules, logistics, and inventories in real time, while sustainability-focused agents continuously evaluate environmental and economic performance.

We envision a future where every engineer manages a team of intelligent agents. This paradigm extends existing workflows and empowers engineers in a scalable, intuitive way. At all times, human engineers remain central to this ecosystem. Rather than replacing human expertise, multi-agent systems are designed to support and augment it. This requires agents to communicate in ways that are transparent, interpretable, and aligned with the users, e.g., through natural language, engineering diagrams, modeling code, and experimental procedures.

Realizing this vision depends on agents being well integrated into the chemical engineering domain. This includes interoperability with modeling and simulation tools, access to validated databases, and the ability to interpret diverse data

modalities. The goal is not to build generic end-to-end black-box AI systems, but domain-specialized teams of agents that understand and respect the intricacies of their subdomain within chemical engineering. The following subsections outline the key building blocks and developments that are required to realize our vision.

8.3.2. ARCHITECTURE

The architecture of a MAS is a key design decision as it substantially influences the conversation workflows, adaptability, and overall performance [12, 15, 54, 55]. Previous research has shown that architectural choices are a major contributing factor to the success of MASs in other domains [26, 54, 56]. For example, [26] proposes a flexible, hierarchical, dynamic, and scalable architecture for code generation and optimization, which yields competitive performance on a domain-specific benchmark. [54] explore four architecture types – centralized, decentralized, and two hybrid architectures – for the task of multi-robot coordination and find that a hybrid architecture yields best results and offers beneficial scaling opportunities for an increasing number of agents. [56] propose AgentForest, a method that repeatedly samples responses from a single language model or a multi-agent framework, and then applies majority voting to select the final answer. The authors compare the performance of AgentForest with common prompt-engineering techniques, multi-agent collaboration approaches, and hybrids that combine those methods with AgentForest [56]. Therefore, identifying suitable MAS architectures for chemical engineering is an important yet open research challenge.

Previous MAS architectures in chemical engineering have mirrored the existing engineering workflows by mapping human and agent roles (cf. Section 8.2.2). Such MAS architectures promote transparency and interpretability while aiming to improve efficiency and enable (semi-)automation. MASs that resemble human-centric workflows are evocative of the concept of expert systems. Expert systems in chemical engineering have a long-standing tradition, dating back over five decades [50]. Expert systems described in literature pre-2000 strongly align with a vision of a transparent and interpretable AI that is grounded in engineering knowledge, for instance, in the cases of physical property prediction [57] and catalyst selection [58]. Such expert systems contain valuable engineering knowledge that should be considered when building MASs. It can be argued that MASs can potentially receive soft engineering knowledge via their architecture if it follows expert systems' behaviours for certain tasks.

An additional approach to following existing engineering workflows could be a modular design that enables agents to coordinate flexibly while maintaining transparency and accountability. This modular design makes the MAS extendable to additional tasks without modifying existing components, unlike end-to-end workflows. In adjacent domains, like quantum chemistry [49] and more distant

domains, like computer science [56, 59], there exist MAS architectures that successfully implement such a modular approach. Some works even use dynamic assembly of agents depending on the task [26]. We recommend exploring how MASs that innovate beyond established workflows compare to those that closely mirror human workflows. In particular, the hybridization between the two approaches could help tap the full potential of MASs in chemical engineering.

8.3.3. TOOL INTEGRATION

Tool integration is critical for MASs in chemical engineering, a domain reliant on specialized software across scales, i.e., from DFT at the molecular level to process simulators like Aspen Plus and gPROMS. Agents must interface with these tools via standardized endpoints (e.g., Python APIs), yet most commercial software lacks such access, posing a key barrier [9]. Therefore, developing open interfaces and data standards is essential. We envision MASs with tool-aware agents: some wrap specific software, while others orchestrate tools for higher-level reasoning. Integration must be built into MAS architectures from the start to ensure scalability, interpretability, and reuse. Future work should examine agent–tool interaction limits, architectural patterns, and strategies for managing tool complexity transparently for both agents and users. Moreover, LLMs might require fine-tuning on our domain modeling languages to take full advantage of modeling environments beyond mere tool calls [60].

8.3.4. DATABASE INTEGRATION

Access to structured, validated data is essential for MASs in chemical engineering, supporting accurate decision-making, reducing hallucinations, and enabling transparent reasoning. However, early MAS efforts have not yet considered database integration, increasing reliance on internal memory or LLM inference, which is prone to errors and unverifiable outputs. We envision MASs interfacing with both open-access and commercial databases across scales: molecular (e.g., QM9, PubChem), reactions (e.g., ORD), thermodynamic (e.g., DIPPR, DDB), and kinetic (e.g., NIST). Besides these databases, the engineering and operational data of plants should also be integrated,e.g., through process historians and Asset Administration Shells. Future work should prioritize developing agents that access these databases via structured APIs or standardized interfaces. Embedding such capabilities early in MAS design will improve modularity, reasoning accuracy, and alignment with engineering practice.

8.3.5. KNOWLEDGE BASE INTEGRATION

Integrating chemical engineering knowledge into MASs is indispensable for reliable decision-making and coordination across tasks and tools. The failure of

general-purpose LLMs at chemical engineering tasks can be attributed to multiple factors, such as inadequate contextual comprehension of domain-specific tokens and jargon, relationships, implicit attributes, and sensitivity to noise in their training data. An open challenge is how to represent and inject existing domain knowledge in a way that remains consistent, traceable, and extensible as systems evolve. A promising approach is to embed a knowledge graph within the MAS to provide a shared, machine-readable schema of concepts, entities, and relations. Domain ontologies such as OntoCape and DEXPI define standardized vocabularies and formal relationships that enable semantic interoperability, provenance tracking, and formal reasoning over domain knowledge. When implemented as knowledge graphs, these ontologies provide a machine-readable data structure that ensures consistency across applications, traceability of data and knowledge, and alignment with domain-specific constraints. The reviewed literature contains the first examples of such knowledge base integration within the chemical engineering domain [39, 44, 46, 48, 51]. However, only one work [39] uses a fixed, domain-specific ontology.

8.3.6. CHEMICAL ENGINEERING FOUNDATION MODELS WITH DOMAIN-SPECIFIC MODALITIES

For MASs to effectively support chemical engineering workflows, the underlying AI must process the field's diverse data modalities (cf. Section 8.2.2). AI cannot assist reliably if it cannot interpret the types of information engineers use daily. Previous approaches to multimodal integration made use of vision language models, e.g., to extract figure descriptions [46, 51] or for the interpretation of scanning electron microscopy images [61]. However, general-purpose and out-of-the-box multimodal LLMs have so far not shown sufficient capability to reliably handle various domain-specific modalities in a chemical engineering context (cf. failure of visual language models in chemistry [62]). Current out-of-the-box LLMs are not sufficiently aware of the chemical engineering context and the domain-specific data modalities. A potential approach could be to tailor domain foundation models via training or fine-tuning. Foundation models, in contrast to LLMs, are designed to perform a wider range of tasks on various modalities beyond text understanding and generation. Chemical engineering foundation models could be used as agents within MASs to help the integration of siloed systems by bridging modality gaps.

We propose developing foundation models trained on chemical engineering–specific multimodal data (cf. Figure 8.3). These models should incorporate multimodal learning, leveraging both labeled and unlabeled datasets through semi-supervised routines. Such semi-supervised learning approaches hold strong potential for fine-tuning in low-data regimes, which are common in chemical

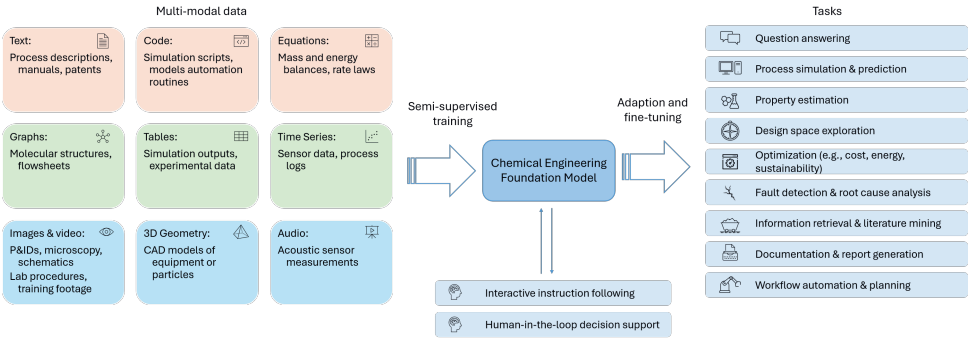


Figure 8.3: Modalities that need to be integrated into a chemical engineering foundation mode. On the left are common modalities and corresponding examples in the chemical engineering domain. A foundation model can be trained in a semi-supervised way. The foundation model can then be customized towards more specific downstream tasks.

engineering [50]. In particular, we envision that graphs will play a key role as a data modality for chemical engineering [63, 64], building on recent advances in computer science [65–69]. Ontology-backed knowledge graphs can further reduce hallucinations through formal constraints and rule-based validation. To get the engineering data in a machine-readable format, digitization efforts may be necessary [70]. For industry use, compact models (e.g., small language models) or federated learning approaches may be necessary.

8.3.7. TRANSPARENCY, SAFETY, AND RESPONSIBLE DEVELOPMENT

MASs must meet the highest standards for transparency, reliability, and accountability, given the safety-critical nature of chemical processes. General-purpose LLMs, while powerful, are prone to hallucinations and lack grounding in physical laws, making them unsuitable as fully autonomous decision-makers in this domain. Specifically, the auto-complete output generation of established encoder-decoder or decoder-only LLMs can cause the propagation of hallucinations in certain application scenarios. Rigorous evaluation of LLM-generated content is always required because outputs that appear coherent and plausible may be incorrect.

Rather than aiming for full automation, MASs should be designed as symbiotic human-AI collaborations. This includes static human-in-the-loop checkpoints and dynamic user consultation during inference - design principles that foster trust, ensure accountability, and align with emerging regulations like the EU AI Act, which mandates human oversight for high-risk AI applications. We consider static monitoring of LLM-generated outputs to be feasible to a certain extent, e.g., type-checking of tool arguments or enforcing limits on repeated tool calls. The

monitoring of LLM outputs that influence safety-critical decisions should always require human approval, e.g., whenever a MAS interfaces with the real world. Even when MASs do not influence high-stakes operation decision-making, the propagation of hallucinations might still cause significant time loss when outputs must be searched for potential errors by human engineers. Previous studies have implemented human-in-the-loop mechanisms, including visual interfaces for user approval [43] and workflows that integrate iterative human feedback [46, 48]. Human involvement can also be included in the training or fine-tuning of LLMs already, e.g., via reinforcement learning with human feedback [71] or through direct preference optimization [72].

MAS architectures must be interpretable and evaluable. While recent open-source benchmark datasets provide a starting point [73, 74], chemical engineering workflows require domain-specific performance criteria and failure taxonomies. Broader adoption will depend on standardized, task-specific evaluation strategies beyond isolated case studies [41, 45]. Interpretable MAS architectures and their design process can be supported through visualization and analysis tools such as AgentCoord [75] or LangChain's LangSmith service. Finally, the development and deployment of LLM-based MASs must consider their computational and environmental costs. Training and operating large models consume significant energy, and these impacts should be assessed using life cycle analysis frameworks to ensure AI workflows support the broader sustainability goals of chemical engineering.

8.4. CONCLUSIONS

MASs are a very recent development in chemical engineering, with only a handful of pioneering studies published to date. Yet, they hold great promise to transform the entire field by essentially providing every engineer with a capable team of specialized agents. We surveyed MASs in chemical engineering, highlighting key developments in architecture, tool and database integration, addressing knowledge integration across modalities, and human-in-the-loop interaction. Despite early promise, the development is only starting, and scientific challenges remain, ranging from designing suitable architectures and integrating tools and data to developing foundation models with domain-specific modalities. In addition, issues of transparency, safety, human oversight, and environmental impact must be addressed. As a young but fast-moving field, MASs offer exciting opportunities to rethink workflows in chemical engineering.

8.5. ACKNOWLEDGEMENT

This research is supported by Shell Global Solutions International B.V. and the Netherlands Organisation for Scientific Research (NWO) under the Veni grant, for which we express sincere gratitude.

REFERENCES

[1] OpenAI, *Gpt-4 technical report,* arXiv preprint arXiv:2303.08774 (2023), arXiv:2303.08774 [cs.CL] .

[2] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, and J. Gao, *Large language models: A survey,* arXiv preprint arXiv:2402.06196 (2024).

[3] A. M. Schweidtmann, *Generative artificial intelligence in chemical engineering,* Nature Chemical Engineering **1**, 193 (2024).

[4] B. Decardi-Nelson, A. S. Alshehri, and F. You, *Generative artificial intelligence in chemical engineering spans multiple scales,* Frontiers in Chemical Engineering **6** (2024), 10.3389/fceng.2024.1458156.

[5] B. Decardi-Nelson, A. S. Alshehri, A. Ajagekar, and F. You, *Generative ai and process systems engineering: The next frontier,* Computers & Chemical Engineering **187**, 108723 (2024).

[6] T. Woo, S. Kim, S. Tariq, S. Heo, and C. Yoo, *Leveraging generative ai and large language model for process systems engineering: A state-of-the-art review,* Korean Journal of Chemical Engineering (2025), 10.1007/s11814-025-00524-y.

[7] P. Daoutidis, J. H. Lee, S. Rangarajan, L. Chiang, B. Gopaluni, A. M. Schweidtmann, I. Harjunkoski, M. Mercangöz, A. Mesbah, F. Boukouvala, F. V. Lima, A. del Rio Chanona, and C. Georgakis, *Machine learning in process systems engineering: Challenges and opportunities,* Computers & Chemical Engineering **181**, 108523 (2024).

[8] V. Venkatasubramanian and A. Chakraborty, *Quo vadis chatgpt? from large language models to large knowledge models,* Computers & Chemical Engineering **192**, 108895 (2025).

[9] W. Du and S. Yang, *The potential and challenges of large language model agent systems in chemical process simulation: from automated modeling to intelligent design,* Frontiers of Chemical Science and Engineering **19** (2025), 10.1007/s11705-025-2587-5.

[10] S. J. Russell, *Artificial intelligence,* third edition, global edition ed., edited by P. Norvig, E. Davis, and D. Edwards, Always learning (Pearson, Boston, 2016) "Authorized adaption from the United States edition, entitled 'Artificial intelligence: a modern approach', third edition, ISBN 9780136042594, by Stuart J. Russell and Peter Norvig published by Pearson Education © 2010." - Rückseite der Titelseite.

[11] M. Wooldridge, *An introduction to multiagent systems* (John wiley & sons, 2009).

[12] B. Liu, X. Li, J. Zhang, J. Wang, T. He, S. Hong, H. Liu, S. Zhang, K. Song, K. Zhu, Y. Cheng, S. Wang, X. Wang, Y. Luo, H. Jin, P. Zhang, O. Liu, J. Chen, H. Zhang, Z. Yu, H. Shi, B. Li, D. Wu, F. Teng, X. Jia, J. Xu, J. Xiang, Y. Lin, T. Liu, T. Liu, Y. Su, H. Sun, G. Berseth, J. Nie, I. Foster, L. Ward, Q. Wu, Y. Gu, M. Zhuge, X. Liang, X. Tang, H. Wang, J. You, C. Wang, J. Pei, Q. Yang,

X. Qi, and C. Wu, *Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems,* (2025), 10.48550/ARXIV.2504.01990, arXiv:2504.01990 [cs.AI] .

[13] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger, and C. Wang, *Autogen: Enabling next-gen llm applications via multi-agent conversation,* (2023), 10.48550/ARXIV.2308.08155, arXiv:2308.08155 [cs.AI] .

[14] T. Liang, Z. He, W. Jiao, X. Wang, Y. Wang, R. Wang, Y. Yang, S. Shi, and Z. Tu, *Encouraging divergent thinking in large language models through multi-agent debate,* (2023), 10.48550/ARXIV.2305.19118, arXiv:2305.19118 [cs.CL] .

[15] K.-T. Tran, D. Dao, M.-D. Nguyen, Q.-V. Pham, B. O'Sullivan, and H. D. Nguyen, *Multi-agent collaboration mechanisms: A survey of llms,* (2025), 10.48550/ARXIV.2501.06322, arXiv:2501.06322 [cs.AI] .

[16] J. He, C. Treude, and D. Lo, *Llm-based multi-agent systems for software engineering: Literature review, vision, and the road ahead,* ACM Transactions on Software Engineering and Methodology **34**, 1 (2025).

[17] A. Ghafarollahi and M. J. Buehler, *Protagents: protein discovery via large language model multi-agent collaborations combining physics and machine learning,* Digital Discovery **3**, 1389 (2024).

[18] D. A. Boiko, R. MacKnight, B. Kline, and G. Gomes, *Autonomous chemical research with large language models,* Nature **624**, 570 (2023).

[19] A. M. Bran, S. Cox, O. Schilter, C. Baldassari, A. D. White, and P. Schwaller, *Augmenting large language models with chemistry tools,* Nature Machine Intelligence **6**, 525 (2024).

[20] Y. Fehlis, C. Crain, A. Jensen, M. Watson, J. Juhasz, P. Mandel, B. Liu, S. Mahon, D. Wilson, N. Lynch-Jonely, B. Leedom, and D. Fuller, *Accelerating drug discovery through agentic ai: A multi-agent approach to laboratory automation in the dmta cycle,* (2025), 10.48550/ARXIV.2507.09023, arXiv:2507.09023 [cs.SE] .

[21] J. Tian, M. T. Sobczak, D. Patil, J. Hou, L. Pang, A. Ramanathan, L. Yang, X. Chen, Y. Golan, X. Zhai, H. Sun, K. Song, and X. Wang, *A multi-agent framework integrating large language models and generative ai for accelerated metamaterial design,* (2025), 10.48550/ARXIV.2503.19889, arXiv:2503.19889 [cond-mat.mtrl-sci] .

[22] J. Gottweis, W.-H. Weng, A. Daryin, T. Tu, A. Palepu, P. Sirkovic, A. Myaskovsky, F. Weissenberger, K. Rong, R. Tanno, K. Saab, D. Popovici, J. Blum, F. Zhang, K. Chou, A. Hassidim, B. Gokturk, A. Vahdat, P. Kohli, Y. Matias, A. Carroll, K. Kulkarni, N. Tomasev, Y. Guan, V. Dhillon, E. D. Vaishnav, B. Lee, T. R. D. Costa, J. R. Penadés, G. Peltz, Y. Xu, A. Pawlosky, A. Karthikesalingam, and V. Natarajan, *Towards an ai co-scientist,* (2025), 10.48550/ARXIV.2502.18864, arXiv:2502.18864 [cs.AI] .

[23] C. Lu, C. Lu, R. T. Lange, J. Foerster, J. Clune, and D. Ha, *The ai scientist: Towards fully automated open-ended scientific discovery,* (2024), 10.48550/ARXIV.2408.06292, arXiv:2408.06292 [cs.AI] .

[24] Y. Yamada, R. T. Lange, C. Lu, S. Hu, C. Lu, J. Foerster, J. Clune, and D. Ha, *The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search,* (2025), 10.48550/ARXIV.2504.08066, arXiv:2504.08066 [cs.AI] .

[25] X. Tang, A. Zou, Z. Zhang, Z. Li, Y. Zhao, X. Zhang, A. Cohan, and M. Gerstein, *Medagents: Large language models as collaborators for zero-shot medical reasoning,* (2023), 10.48550/ARXIV.2311.10537, arXiv:2311.10537 [cs.CL] .

[26] Y. Ishibashi and Y. Nishimura, *Self-organized agents: A llm multi-agent framework toward ultra large-scale code generation and optimization,* (2024), 10.48550/ARXIV.2404.02183, arXiv:2404.02183 [cs.SE] .

[27] M. R. Boskabadi, Y. Cao, B. Khadem, W. Clements, Z. Nevin Gerek, E. Reuthe, A. Sivaram, C. J. Savoie, and S. S. Mansouri, *Industrial agentic ai and generative modeling in complex systems,* Current Opinion in Chemical Engineering **48**, 101150 (2025).

[28] S. Rothfarb, M. Friday, X. Wang, A. Zaghi, and B. Li, *Multi-agent large language model frameworks: Unlocking new possibilities for optimizing wastewater treatment operation,* Environmental Research **275**, 121401 (2025).

[29] Z. Hong, Z. Yuan, Q. Zhang, H. Chen, J. Dong, F. Huang, and X. Huang, *Next-generation database interfaces: A survey of llm-based text-to-sql,* (2024), 10.48550/ARXIV.2406.08426, arXiv:2406.08426 [cs.CL] .

[30] M. G. Ozsoy, L. Messallem, J. Besga, and G. Minneci, *Text2cypher: Bridging natural language and graph databases,* (2024), 10.48550/ARXIV.2412.10064, arXiv:2412.10064 [cs.LG] .

[31] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, *Retrieval-augmented generation for knowledge-intensive nlp tasks,* (2020), 10.48550/ARXIV.2005.11401, arXiv:2005.11401 [cs.CL] .

[32] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, and J. Larson, *From local to global: A graph rag approach to query-focused summarization,* (2024), 10.48550/ARXIV.2404.16130, arXiv:2404.16130 [cs.CL] .

[33] H. Han, Y. Wang, H. Shomer, K. Guo, J. Ding, Y. Lei, M. Halappanavar, R. A. Rossi, S. Mukherjee, X. Tang, Q. He, Z. Hua, B. Long, T. Zhao, N. Shah, A. Javari, Y. Xia, and J. Tang, *Retrieval-augmented generation with graphs (graphrag),* (2025), 10.48550/ARXIV.2501.00309, arXiv:2501.00309 [cs.IR] .

[34] Y. Hu, Z. Lei, Z. Zhang, B. Pan, C. Ling, and L. Zhao, *Grag: Graph retrieval-augmented generation,* (2024), 10.48550/ARXIV.2405.16506, arXiv:2405.16506 [cs.LG] .

[35] I. CrewAI™, (2025).

[36] L. Inc, *Langgraph,* .

[37] G. Li, H. A. A. K. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem, *Camel: Communicative agents for "mind" exploration of large language model society,* (2023), 10.48550/ARXIV.2303.17760, arXiv:2303.17760 [cs.AI] .

[38] S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, C. Zhang, J. Wang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, C. Ran, L. Xiao, C. Wu, and J. Schmidhuber, *Metagpt: Meta programming for a multi-agent collaborative framework,* (2023), 10.48550/ARXIV.2308.00352, arXiv:2308.00352 [cs.AI] .

[39] N. P. Taskiran, C.-E. J. Tsai, S. Huang, A. Chakraborty, and V. Venkatasubramanian, *A knowledge-graph-based pharmaceutical engineering chatbot for drug discovery,* Computers & Chemical Engineering **203**, 109318 (2025).

[40] S. Han, Q. Zhang, Y. Yao, W. Jin, and Z. Xu, *Llm multi-agent systems: Challenges and open problems,* (2024), 10.48550/ARXIV.2402.03578, arXiv:2402.03578 [cs.MA] .

[41] D. Lee, J. Lee, and D. Shin, *Gpt prompt engineering for a large language model-based process improvement generation system,* Korean Journal of Chemical Engineering **41**, 3263 (2024).

[42] J. Vyas and M. Mercangöz, *Autonomous industrial control using an agentic framework with large language models,* (2024), 10.48550/ARXIV.2411.05904, arXiv:2411.05904 [cs.MA] .

[43] S. Gowaikar, S. Iyengar, S. Segal, and S. Kalyanaraman, *An agentic approach to automatic creation of p&id diagrams from natural language descriptions,* (2024), 10.48550/ARXIV.2412.12898, arXiv:2412.12898 [cs.LG] .

[44] S. S. Sakhinana, V. Sri Vaikunth, and V. Runkana, *Knowledge graph modeling-driven large language model operating system (llm os) for task automation in process engineering problem-solving,* Proceedings of the AAAI Symposium Series **4**, 222 (2024).

[45] E. Pajak, A. Bahamdan, K. Hellgardt, and A. d. Río-Chanona, *Multi-agent llms for automating sustainable operational decision-making,* in *Proceedings of the 35th European Symposium on Computer Aided Process Engineering (ESCAPE 35)*, ESCAPE 35, Vol. 4 (PSE Press, 2025) pp. 1824–1829.

[46] S. S. Srinivas, S. Gupta, and V. Runkana, *Autochemschematic ai: A closed-loop, physics-aware agentic framework for auto-generating chemical process and instrumentation diagrams,* (2025), 10.48550/ARXIV.2505.24584, arXiv:2505.24584 [cs.LG] .

[47] T. Zeng, S. Badrinarayanan, J. Ock, C.-K. Lai, and A. B. Farimani, *Llm-guided chemical process optimization with a multi-agent approach,* (2025), 10.48550/ARXIV.2506.20921, arXiv:2506.20921 [cs.LG] .

[48] S. S. Srinivas, A. Das, S. Gupta, and V. Runkana, *Accelerating manufacturing scale-up from material discovery using agentic web navigation and retrieval-augmented ai for process engineering schematics design,* (2024), 10.48550/ARXIV.2412.05937, arXiv:2412.05937 [cs.LG] .

[49] Y. Zou, A. H. Cheng, A. Aldossary, J. Bai, S. X. Leong, J. A. Campos-Gonzalez-Angulo, C. Choi, C. T. Ser, G. Tom, A. Wang, Z. Zhang, I. Yakavets, H. Hao, C. Crebolder, V. Bernales, and A. Aspuru-Guzik, *El agente: An autonomous agent for quantum chemistry,* Matter **8**, 102263 (2025).

[50] V. Venkatasubramanian, *The promise of artificial intelligence in chemical engineering: Is it here, finally?* AIChE Journal **65**, 466 (2018).

[51] D. Ovalle, A. Seth, J. R. Kitchin, C. D. Laird, and I. E. Grossmann, *Grapse: Graph-based retrieval augmentation for process systems engineering,* in *Proceedings of the 35th European Symposium on Computer Aided Process Engineering (ESCAPE 35)*, ESCAPE 35, Vol. 4 (PSE Press, 2025) pp. 1598–1604.

[52] A. M. Schweidtmann, E. Esche, A. Fischer, M. Kloft, J.-U. Repke, S. Sager, and A. Mitsos, *Machine learning in chemical engineering: A perspective,* Chemie Ingenieur Technik **93**, 2029 (2021).

[53] A. A. Alimin, D. P. Goldstein, L. S. Balhorn, and A. M. Schweidtmann, *Talking like piping and instrumentation diagrams (p&ids),* (2025), 10.48550/ARXIV.2502.18928, arXiv:2502.18928 [cs.AI] .

[54] Y. Chen, J. Arkin, Y. Zhang, N. Roy, and C. Fan, *Scalable multi-robot collaboration with large language models: Centralized or decentralized systems?* The 2024 International Conference on Robotics and Automation (2023), 10.48550/ARXIV.2309.15943, arXiv:2309.15943 [cs.RO] .

[55] J. Fang, Y. Peng, X. Zhang, Y. Wang, X. Yi, G. Zhang, Y. Xu, B. Wu, S. Liu, Z. Li, Z. Ren, N. Aletras, X. Wang, H. Zhou, and Z. Meng, *A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems,* (2025), 10.48550/ARXIV.2508.07407, arXiv:2508.07407 [cs.AI] .

[56] J. Li, Q. Zhang, Y. Yu, Q. Fu, and D. Ye, *More agents is all you need,* (2024), 10.48550/ARXIV.2402.05120, arXiv:2402.05120 [cs.CL] .

[57] R. Bañares-Alcántara, A. W. Westerberg, and M. D. Rychener, *Development of an expert system for physical property predictions,* Computers & Chemical Engineering **9**, 127 (1985).

[58] R. Banãres-Alcántara, E. Ko, A. Westerberg, and M. Rychener, *Decade—a hybrid expert system for catalyst selection—ii. final architecture and results,* Computers & Chemical Engineering **12**, 923 (1988).

[59] W. Chen, Y. Su, J. Zuo, C. Yang, C. Yuan, C.-M. Chan, H. Yu, Y. Lu, Y.-H. Hung, C. Qian, Y. Qin, X. Cong, R. Xie, Z. Liu, M. Sun, and J. Zhou, *Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors,* (2023), 10.48550/ARXIV.2308.10848, arXiv:2308.10848 [cs.CL] .

[60] S. Rupprecht, Y. Hounat, M. Kumar, G. Lastrucci, and A. M. Schweidtmann, *Text2model: Generating dynamic chemical reactor models using large language models (llms),* arXiv preprint arXiv:2503.17004 (2025).

[61] A. S. Alshehri, F. L. Lee, and S. Wang, *Multimodal deep learning for scientific imaging interpretation,* (2023), 10.48550/ARXIV.2309.12460, arXiv:2309.12460 [cs.LG] .

[62] N. Alampara, M. Schilling-Wilhelmi, M. Ríos-García, I. Mandal, P. Khetarpal, H. S. Grover, N. M. A. Krishnan, and K. M. Jablonka, *Probing the limitations of multimodal language models for chemistry and materials research,* (2024), 10.48550/ARXIV.2411.16955, arXiv:2411.16955 [cs.LG] .

[63] Q. Gao and A. M. Schweidtmann, *Deep reinforcement learning for process design: Review and perspective,* Current Opinion in Chemical Engineering **44**, 101012 (2024).

[64] L. S. Balhorn, K. Degens, and A. M. Schweidtmann, *Graph-to-sfiles: Control structure prediction from process topologies using generative artificial intelligence,* Computers & Chemical Engineering , 109121 (2025).

[65] S. Ma, C. Xu, X. Jiang, M. Li, H. Qu, C. Yang, J. Mao, and J. Guo, *Think-on-graph 2.0: Deep and faithful large language model reasoning with knowledge-guided retrieval augmented generation,* (2024), 10.48550/ARXIV.2407.10805, arXiv:2407.10805 [cs.CL] .

[66] B. Perozzi, B. Fatemi, D. Zelle, A. Tsitsulin, M. Kazemi, R. Al-Rfou, and J. Halcrow, *Let your graph do the talking: Encoding structured data for llms,* (2024), 10.48550/ARXIV.2402.05862, arXiv:2402.05862 [cs.LG] .

[67] B. Jin, G. Liu, C. Han, M. Jiang, H. Ji, and J. Han, *Large language models on graphs: A comprehensive survey,* Transactions on Knowledge and Data Engineering (TKDE) 2024 (2023), 10.48550/ARXIV.2312.02783, arXiv:2312.02783 [cs.CL] .

[68] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, *Unifying large language models and knowledge graphs: A roadmap,* IEEE Transactions on Knowledge and Data Engineering (TKDE) 2024 **36**, 3580 (2023), arXiv:2306.08302 [cs.CL] .

[69] L. Luo, Y.-F. Li, G. Haffari, and S. Pan, *Reasoning on graphs: Faithful and interpretable large language model reasoning,* (2023), 10.48550/ARXIV.2310.01061, arXiv:2310.01061 [cs.CL] .

[70] M. F. Theisen, K. N. Flores, L. Schulze Balhorn, and A. M. Schweidtmann, *Digitization of chemical process flow diagrams using deep convolutional neural networks,* Digital Chemical Engineering **6**, 100072 (2023).

[71] P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, *Deep reinforcement learning from human preferences,* (2017), 10.48550/ARXIV.1706.03741, arXiv:1706.03741 [stat.ML] .

[72] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn, *Direct preference optimization: Your language model is secretly a reward model,* (2023), 10.48550/ARXIV.2305.18290, arXiv:2305.18290 [cs.LG] .

[73] M. Cemri, M. Z. Pan, S. Yang, L. A. Agrawal, B. Chopra, R. Tiwari, K. Keutzer, A. Parameswaran, D. Klein, K. Ramchandran, M. Zaharia, J. E. Gonzalez, and I. Stoica, *Why do multi-agent llm systems fail?* (2025), 10.48550/ARXIV.2503.13657, arXiv:2503.13657 [cs.AI] .

[74] F. Grötschla, L. Müller, J. Tönshoff, M. Galkin, and B. Perozzi, *Agentsnet: Coordination and collaborative reasoning in multi-agent llms,* (2025), 10.48550/ARXIV.2507.08616, arXiv:2507.08616 [cs.MA] .

[75] B. Pan, J. Lu, K. Wang, L. Zheng, Z. Wen, Y. Feng, M. Zhu, and W. Chen, *Agentcoord: Visually exploring coordination strategy for llm-based multi-agent collaboration,* (2024), 10.48550/ARXIV.2404.11943, arXiv:2404.11943 [cs.HC] .

VI

SUMMARY AND LOOKING AHEAD

9

CONCLUSIONS AND OUTLOOK

The preceding chapters developed graph-based methods for environmental impact prediction, polymer property prediction under label scarcity, Bayesian uncertainty quantification, RL for flowsheet synthesis, and multi-agent orchestration of these components. This final chapter reflects on what has been learned at each of these levels, revisits the two objectives and five challenges that motivated the thesis, and discusses the limitations of the approach and the open questions the author considers most consequential for continued work in this direction.

9.1. ON THE SUITABILITY OF GRAPH REPRESENTATIONS

Treating molecules and processes as graphs is advantageous both conceptually and operationally. Conceptually, the node–edge abstraction aligns with the atom–bond pairs of molecules and the unit–stream connections of process flowsheets, allowing models to respect the locality and topology that govern molecular properties and flowsheet behavior. Operationally, node and edge embeddings provide a flexible mechanism for encoding the detailed structural and specification-level information that characterizes both objects. The distinction in connectivity is itself informative: largely undirected edges in molecules focus the representation on chemical neighborhoods and scaffold patterns, whereas directed edges in processes encode the inherent orientation of streams and recycles.

A further and often under-appreciated advantage is permutation invariance. Neither molecules nor flowsheets possess a canonical ordering of their components, and representations that impose one, such as SMILES strings or sequentially indexed unit lists, force the model to relearn equivalences. Graph neural networks, by construction, produce identical outputs under any permutation of nodes, which removes this burden and leaves model capacity free for the properties of genuine chemical or topological interest. Together, these alignments allow message passing to aggregate physically meaningful context and enable graph-level readouts to serve as stable fingerprints for prediction and optimization.

However, the case at the level of the integration workflow is less settled. This thesis proposes modeling the workflow as a graph of interacting agents and information flows, and this view is useful for reasoning about how specialized tools should be coordinated. Whether it is the most suitable representation, however,

remains an open question. The agents and information flows involved at this level are heterogeneous in ways that molecular and process graphs are not, including the irregular involvement of human experts, the dynamic creation and dissolution of agent connections, and the mixture of structured and unstructured information being exchanged. Alternative formalisms, such as hierarchical task networks, may turn out to be better suited to capturing these dynamics. The position of this thesis is that the graph abstraction is a starting point for the integration layer, not that it is necessarily the final answer.

9.2. GRAPH MACHINE LEARNING FOR MOLECULAR PROPERTIES

The three contributions on the molecular side cover environmental impact prediction, self-supervised pretraining for polymers, and Bayesian uncertainty quantification.

The first is that GNNs are a viable alternative to QSPR models for predicting impact categories. On the CarbonMinds dataset, GNNs improved climate-change prediction over QSPR baselines, and multi-task formulations exploited correlations across categories. Land use, ionising radiation, and fossil resource depletion, however, remained poorly predicted regardless of architecture, because their variance is driven less by molecular structure than by where and how a molecule is produced. GNN models with molecular features are therefore best understood as an early-stage screening tool, and any rigorous LCA prediction system will need to augment molecular graphs with process-level information.

Second, self-supervised pretraining is an effective method for label scarcity in polymer property prediction, improving R^2 by more than 30% under extreme label scarcity on the benchmark studied. The more instructive finding concerns which parameters carry useful inductive bias: the informative representation from pretraining is preserved not only in the encoder but also in the fully connected layers, and transferring the full stack of fully connected layers gives the most robust fine-tuning performance. These findings, however, are based on a single computational benchmark. Whether the advantage of pretraining, and in particular the benefit of transferring the fully connected layers, holds across other polymer datasets, properties, and pretraining corpora remains to be established. The recommended strategy of pretraining on a large unlabeled dataset, transferring the full network, and fine-tuning should therefore be understood as a well-supported hypothesis for further testing rather than a general rule.

Third, Bayesian uncertainty quantification on GNNs is feasible, and the results indicate that it can be competitive with the ensemble and dropout methods that currently dominate the literature. On the benchmark studied, Parallel SGHMC produced sharper and better calibrated uncertainty estimates than Deep Ensembles and MC Dropout. This evidence should, however, be regarded as preliminary:

the evaluation covers a single QM9 property on a single data split, does not report the variability of performance across repeated runs, and compares against a Deep Ensemble with a limited number of members. A stronger ensemble baseline may narrow or close the observed gap, so a firm conclusion on the relative merit of SGHMC requires the broader evaluation discussed in the limitations below. Independently of which method ultimately proves superior, the more general lesson is that uncertainty quantification should be treated as a requirement rather than a diagnostic added after the fact. Claims about its value in high-stakes settings, such as pharmaceutical development, must be supported by evaluations considerably broader than the one presented here.

9.3. GRAPH MACHINE LEARNING FOR PROCESS DESIGN

RL is a promising method for flowsheet synthesis, but its learning efficiency remains the binding constraint, since each training episode requires a call to a rigorous process simulator. Transfer learning offers a direct remedy. Pretraining on shortcut simulators and transferring the encoder and actor to the rigorous setting cut convergence time by roughly 64% against the no-transfer baseline, while marginally improving final scores. Structural priors from process VAE latents gave complementary but smaller gains. The takeaway is that targeted transfer is an effective accelerator for RL-based synthesis, and that graph-structured RL is a viable approach for problems with discrete topology, soft constraints, and noisy simulators, where deterministic superstructure methods cannot readily be posed or solved.

The transfer study also produced a negative result worth emphasizing. Reusing continuous-action or critic networks across environments with mismatched action ranges or reward shaping produced negative transfer, and encoder pretraining on legacy flowsheet data gave only marginal gains unless source-task complexity matched the target. Transfer learning in this setting is therefore a carefully scoped accelerator, not a general approach for sample inefficiency. The practical recipe is therefore to transfer the early representation layers and the actor head, but to reinitialize action and value networks whenever dynamics or rewards differ.

9.4. GRAPH MACHINE LEARNING FOR ORCHESTRATION

MASs were proposed as an orchestration layer over the preceding GNN- and RL-based tools, but this contribution is a perspective rather than a validated result. The survey identified the required building blocks, namely agent architectures, tool integration, knowledge management, and human-in-the-loop interaction, but did not show that such a system outperforms either human engineers or the underlying tools on an end-to-end design task. What the chapter offers is a

concrete target: a framework in which molecular and process agents are coordinated together, and in which human oversight is structurally embedded. Whether multi-agent orchestration delivers on this target is, at the time of writing, an open question.

9.5. RETURNING TO THE TWO OBJECTIVES AND FIVE CHALLENGES

This thesis set out with a twofold objective. The first was to develop a set of deep learning tools that improve on existing methods for molecular property prediction and process design. The second was to compose these tools, together with existing ones, into an integrated system that supports the engineer across the design workflow rather than at isolated steps within it. Concrete progress was made in the first level through the molecular and process contributions summarized in the preceding sections. The second level of the objective was presented as a perspective rather than a validated system: A multi-agent orchestration layer over the developed tools was designed and surveyed, but its empirical evaluation against engineer-led workflows remains future work.

The five challenges that motivated this work are addressed as follows. Data scarcity was addressed through self-supervised pretraining on polymer corpora and through fidelity-based transfer learning on process simulators. The data representation challenge was answered by adopting graphs as the shared substrate across molecular, process, and workflow scales, with permutation invariance and topology preservation as the principal benefits. Calibrated uncertainty was addressed through Parallel SGHMC, which showed improvements in sharpness and calibration over Deep Ensembles and MC Dropout on the benchmark studied, although broader validation is still required. Optimal decision-making was reformulated as sequential construction on a directed graph and solved by hierarchical RL. The integration challenge was framed and partially addressed through the multi-agent orchestration framework, which provides a concrete target architecture even where empirical demonstration remains open. The overarching contribution is not any single method but the demonstration that graph-based learning, pursued consistently across scales, can serve as the connective tissue that begins to close the gap between molecular and process design.

9.6. LIMITATIONS

While this dissertation advances the use of graph machine learning in chemical engineering, several limitations remain and point toward necessary future developments.

First, regarding molecular modeling, the frameworks proposed herein are

designed to complement, rather than replace, rigorous first-principles simulation. While our graph-based models accelerate screening by proposing high-probability candidates, final validation must still rely on detailed thermodynamic and kinetic modeling. The LCA-prediction framework, as noted above, is fundamentally limited by its reliance on molecular structure alone.

Second, regarding process design, a fundamental methodological tension persists in applying RL to what is inherently a static optimization problem. Mapping the simultaneous conceptual design task into a sequential construction process introduces significant complexities in state representation and reward shaping. Consequently, the sample efficiency of this approach compared to established deterministic methods (such as Superstructure Optimization or MINLP) remains a critical bottleneck. Closely related to this is the challenge of generalization: while our agents demonstrate design capability within their training distributions, their ability to extrapolate to unseen design specifications, novel unit operations, or distinct chemical systems (without retraining) remains an open challenge that this work characterizes but does not fully solve.

Third, regarding the orchestration of these tools via MASs, we face the challenges of reliability and autonomy. Current LLMs still struggle with long-horizon reasoning and hallucination, which pose severe risks in safety-critical engineering contexts. As these systems move toward autonomy, critical issues regarding transparency, human oversight, and the verification of generated designs must be rigorously addressed.

Beyond these methodological limitations, the strength of the empirical evidence differs between chapters. The environmental impact study (Chapter 4) and the transfer learning study for flowsheet synthesis (Chapter 7) are supported by comparatively extensive evaluations. By contrast, the pretraining study for polymers (Chapter 5) relies on a single computational benchmark, and the uncertainty quantification study (Chapter 6) evaluates a single QM9 property on a single data split, without reporting performance variability across repeated runs, and against a Deep Ensemble baseline with a limited number of members. The conclusions of these two chapters should therefore be read as preliminary evidence rather than general statements. A more rigorous assessment would include multiple datasets and properties, repeated data splits with reported error bars, and baselines of comparable computational budget, for example Deep Ensembles of varying size. Furthermore, the uncertainty estimates produced by SGHMC were only assessed within the training distribution. Detecting inputs outside that distribution remains a separate problem that this thesis does not address.

9.7. OUTLOOK

The directions outlined here are scoped to what this dissertation leaves open, and to what a researcher continuing this line of work would plausibly take up next.

For molecular models, the three contributions should be combined into a single framework for molecular property prediction: a GNN pretrained via self-supervision on an unlabeled molecular dataset, fine-tuned in a multi-task setting on the available labels, and equipped with Bayesian uncertainty quantification over its outputs. Each component has been studied in isolation in this dissertation, with varying depth of validation, and combining them should go together with the broader benchmarking called for above. The integration would yield a general tool applicable across property prediction domains, from environmental impact and thermodynamic properties to drug discovery, catalysis, and beyond.

For process design, the most important open question concerns inference rather than training. It has been demonstrated that an RL agent can learn to design flowsheets within its training distribution, but its ability to generalize to unseen specifications at inference time remains unresolved. Progress here requires agents that can transfer their learned policies zero-shot or with minimal adaptation, and architectures that expose the structural regularities of the design space rather than memorizing them. A tangible starting point would be to train the agent on a family of separation tasks with varying feed compositions and then evaluate, without any fine-tuning, on feed compositions held out from training, measuring how design quality degrades as the test specifications move further from the training distribution.

For orchestration, the next concrete step is a framework that couples the molecular and process tools developed in this dissertation into a design workflow: the unified molecular predictor identifies candidate molecules with quantified uncertainty, the RL agent generates flowsheets under those uncertainties, and the multi-agent layer routes information between them under human oversight. A tractable first version would restrict the scope to a single, well-defined design problem, such as selecting a green solvent and synthesizing its corresponding separation process, and measure whether the coupled system reaches a valid end-to-end design with tools. Building and evaluating such a framework is the most consequential open problem this dissertation points toward.

Beyond these concrete next steps, several longer-term scientific priorities emerge from this work. First, the field needs shared, openly available benchmarks for chemical engineering tasks, together with evaluation standards that report variability across data splits and compare methods at equal computational budget. Much of the uncertainty about which methods genuinely work, including in this thesis, stems from the absence of such standards. Second, data scarcity is unlikely to be solved by individual research groups. Foundation models pretrained on

large collections of molecular, polymer, and process data, possibly shared between academia and industry through approaches that protect proprietary data such as federated learning, appear to be the most promising route. Third, purely data-driven models should increasingly be combined with physical knowledge, through hybrid models that respect conservation laws and thermodynamic consistency, which would improve both extrapolation and trust. Fourth, closing the loop between computation and experiment, by coupling generative design and uncertainty-aware prediction with automated laboratories, would allow models to select the experiments from which they learn most. Finally, as AI systems take on a larger role in design decisions, verification, interpretability, and accountability will become scientific priorities in their own right rather than afterthoughts.

Looking further ahead, the contributions of this thesis point toward a future in which AI is not an auxiliary tool invoked at specific steps of the design workflow but the substrate on which the workflow itself runs. In this future, molecular discovery, property prediction, process synthesis, and techno-economic and environmental assessment are no longer separate activities performed by different software or tools. On such a platform, the system would formulate the task, investigate the underlying problem, draw on relevant tools, databases, and models, and query human experts when necessary. The engineer's role shifts accordingly, from constructing models and executing simulations to specifying objectives, interrogating trade-offs, and adjudicating between candidate designs that the system itself has generated and evaluated.

ACKNOWLEDGEMENTS

Time is a funny thing. Over five years, colleagues and friends sitting in the same or different offices slowly turn into a book, into a photo. Now it is my turn to write the closing chapter of my own book.

First and foremost, I want to thank Artur. In my propositions, I wrote: "The quality of supervision, rather than the topic itself, is the strongest determinant of the PhD experience." This basically describes my whole PhD life. You are approachable as a friend, informative and inspiring as a supervisor, supportive and encouraging as a therapist, and caring and considerate as a proper academic parent. Lukas and I have constantly talked about how lucky we are to be your first two PhD students. You helped, guided and saved us throughout this five year journey. I am always surprised by how much energy you have, and how you pour all of it into your work with such passion. I am surprised by how many encouraging words you have given us along the way, and by how much effort you have put into making us an actual team. I would also like to thank Jana. Thank you for your enormous help throughout my PhD. Your guidance and positive attitude still shape how I approach everything.

Lukas, honestly, I don't know how to write this. For the past five years, we shared sweet moments, and bitter times stuck at bottlenecks in both bouldering and research. We shared cake, we shared food, we shared laziness about writing, we shared the struggle of finding meaning in life, we shared the excitement of every little improvement in climbing, we shared the anxiety of graduating, we shared the happiness of crazy conference after parties, we shared spicy gossip, we shared friends, we shared the same bikes, trains, flights and boats (one called Anthena), we shared the same office and even the same house, we shared the same rope on a 20 meter fake wall in the Netherlands, and we also shared the same rope on a 200 meter, 7 pitch climb in the Alps (not trying to show off here). We shared our whole PhD life, and remarkably, we even share the same graduation day. There are so many unforgettable memories and highlights from these five years, and most of them are because of you.

Giacomo, the first time I "met" you was through a photo. I said to Lukas: holy crap, why does this guy look like a model? Later, when I actually got to know you, I realized that elegance can be a way of living, not just an adjective. You and Marta helped me understand Tuscany: understated, vintage, warm without being loud. Thank you for showing me that such a way of living exists. I wish you and Marta all the best.

Max, you are one of the few people who can be funny and deep at the same time, and who also happens to love terrible jokes. At the start, I was your master thesis supervisor, but honestly, I often wondered what guidance I could possibly

give you. Later you joined as a PhD, and working with you made me realize that being able to share ideas with someone and instantly click is something precious in itself. The Erasmus+ weeks were some of the happiest of my PhD: the mountain roads in Slovakia, the driving in Sicily, and you drinking all of my wine. I hope we can go on another trip together someday.

Tanuj, you might be the most extroverted person I have ever met, like a warm lamp in winter that brings its own heat and energy. I often quietly envied people like you, who can be so natural, passionate and genuine with everyone in any conversation. Knowing you made me rethink what being extroverted really means: being present, and being willing to give yourself to others. I hope we meet again, my friend.

Dominik, you are probably one of the most unique people I have ever met. On one side, you are a diplomat: decent, careful and elegant. On the other side, you are a child: chaotic, obsessive and endlessly curious. Having both sides at once is what makes you so special. Thank you for showing me that a person can be so completely themselves. All the best, my friend.

Sophia, you are probably the most thoughtful, careful and considerate person I have ever known. Max once said the whole group dynamic changed after you joined, and I completely agree. You brought a softness and warmth that the group never had before. I hope you can feel relaxed and free of stress, every day and all the time.

Angga, the first time we met was online, and I still remember thinking you seemed like a really chill guy. After getting to know you in person, I can confirm it: you are one of the most relaxed and easygoing people I know, and being around you always made me feel calm. Thank you for all the local Indonesian food you brought to us. Every time, it was insanely delicious, and every time, it disappeared in minutes. I wish you all the best, my friend.

Ulderico, although we have not known each other for long, you have left a strong impression on me. You are funny, witty and at the same time very professional, which is a rare combination. You are just like the sunshine of Napoli: warm, bright, open and full of passion, and you bring that energy into every room you walk in. Thank you for all the laughs and the good conversations. I wish you all the best.

Marco, although we met quite late, it did not take long for us to become good office coffee buddies. You are one of the most easygoing people I have met, and talking with you always felt natural and relaxed. Our coffee breaks became one of the small things I looked forward to during the last part of my PhD. Thank you for the good company, and I wish you all the best.

Rens, I really enjoyed our conversations about music and movies. You always have sharp and unique opinions, and you are direct and sincere in a way I truly

appreciate. I hope we can keep sharing music with each other, haha.

Christian, you have a unique kind of Dutch elegance and decency. I still remember our first Christmas together, when we finished the paella a whole hour earlier than planned, and the time we drove out at noon just to get KFC. I wish you a calm and happy life in "Germany".

Nasim, I really admire your drive and perseverance. I am often amazed by how determined and focused you are in both your studies and your work. I hope you achieve everything you are aiming for.

Lena, I am always amazed by your incredible energy and your contagious smile. Thank you for hosting one cocktail party after another. I wish you laughter every single day.

Riccardo, you are basically a walking encyclopedia. I am still amazed that someone can be so knowledgeable and so funny at the same time. I wish you a perfect life.

Marc, you are unbelievably optimistic and so much fun, haha. You are just like the weather in Barcelona: sunny, bright and always easygoing. I wish you all the best.

Gabriel, we have known each other for a long time, and you are ridiculously handsome. Thank you for inviting me to every one of your parties. I hope you are happy every day, and I wish you all the best.

Laura, you were probably my very first collaborator during my PhD. I was often amazed by how someone could be so smart, so organized and so fun at the same time. I wish you a smooth and successful PhD as well.

Thank you to everyone in PPE and PSE: Isabell, Albert, Ruben, Zaid, Pranav, Bhavesh, Simone, Sophie, Kalani, Dorian, Ignasi, Hardlife, Athina, Gurhan, Margherita, Emma, Ida, Ming, Taeyong, Zhimian, Adarsh, Susanne, Lucas, Andreano and Lorenzo. It is truly rare to find such an amazing atmosphere and safe environment during a PhD. This experience has shaped and changed me in so many ways, and I am grateful to everyone I met.

I would like to express my sincere gratitude to my master's thesis supervisor, Andy Keller. I have never met anyone quite like you: intelligent, empowering, patient and sincere. I could not have made it through without your guidance and help during the COVID period, and I will always remember everything you taught me back then. Best wishes to you, my teacher and my friend.

I would also like to thank all the students, researchers, friends and collaborators I have worked with. None of my PhD work would have gone so smoothly without your contribution

李晴，作为整个系为数不多的中国人，非常享受我们用中文当着别人面聊八卦的过程。你非常的真诚和聪明。希望你能永远健康，天天开心！还有恺乔，怀念我们一起约饭，一起吐槽的时候，有空去广州找你约饭啊！！

哈喽斌洁，不知不觉也认识你七年了。第一天认识你我就觉得和你超级合拍。和你一起经历了很多，旅游，谈心，聊八卦，聊一切。你真的蜕变了很多，希望你知道，你真的很厉害，很坚强，很值得。你并不需要别人的认可来满足自己，因为你一直都很棒！希望我们都能得偿所愿！！

同时我也想感谢其他朋友们，王帅，钱煜，孙文博，李玫芳，硕士期间一起吐槽，约饭。也感谢斌哥，红姐，等等攀岩友们。五年时光多姿多彩。

对父母，我还记得25年回去的时候，凌晨坐在老爸老妈的车里，老爸说:"从小学初中天天接送你上下学，到高中一个星期一次，再到大学的半年一次，到如今我们一年接你一次"。当时听到这时颇有感触，是啊，可能人生就是不断的远离父母，然后短暂回归，然后重置。只是每次回来，我也许也慢慢长大，而褶皱和白发却也硬生生的烙刻在你们俩的眼纹和鬓角。而我能做的，也只能许你们所谓的世俗的成功，给你们些许慰藉。陪伴是个永恒的话题，尤其对海外游子，我会尽我所能，但结局大多是亏欠和遗憾。时间尺度虽然是匀速前行，但人生的感觉和感悟却是有着轻重缓急。欲买桂花同载酒，终不似，少年游。对姐姐，从我记事起，我们就住在一起，从租的平方，到买的小区，再到高中，你从这个家庭驶向了另一个家庭。因为沟通的缺失，我们从未真正的深切交流过，大家都以为心知肚明对彼此的感情。感谢你们，我爱你们。

七年前，启程的前一晚，我和妈妈在老家吃完饭出去散步，老爸那时在出差。村里的灯光不算敞亮，我走得稍慢了些，对着昏暗的灯光，拍下了老妈的背影。手机的夜拍不太行，照片拍糊了，但我看了一眼，还挺喜欢的，挺有艺术感。七年后的今天，我把它当作了我博士论文的封面。一切都从那一刻，延伸到了此刻。

刚落地的时候，不知是因为时差还是兴奋，一切都很新鲜：湛蓝的天空，糟糕的厨艺，蹩脚的英语。可我是如此高兴，因为出国是我从高中起就有的梦想。

硕士刚开始的时候，学业是如此艰难：听不懂的课堂，说不流利的英语，写不出的代码。可我却从未如此快乐过，因为那段时光是那么纯粹。在自己选择的路上，孜孜不倦地汲取知识，探索边界，也探索人生。

即使是疫情期间，一切都被隔断，大家只能在网上相见，我却在那段时间如鱼得水，自由徜徉。

博士这五年更像是一场梦。不知道是读博养成的，还是我本来的习惯，我常常把自己抽离出来，用旁观者的视角审视自己。第一次去葡萄牙参加夏校，傍晚我们一群人踏着凉爽的空气走回住处，暗蓝色铺满整个天际，路边的灯光影影绰绰。我就在想：我这是在做梦吗？我何德何能，走到了这里？

这个问题一直萦绕在我脑海里。在雅典的会议晚宴上，大家纵情畅饮，我也醉得跳起舞来时；在希腊的游船上，咸咸的海风把头发吹得黏在一起时；在奥地利的阿尔卑斯山上，七段的攀岩爬到力竭时；更是在冰岛那片人世间的尽头，站在我毕生梦想之地时。

9

也许人真的就活在这些瞬间里。我很庆幸，我拥有过这些瞬间，也相信我会继续追逐这些瞬间。

人生是一场关于自己的修行。七年前，我从没想过会成为现在的我。第一次鼓起勇气，第一次心血来潮，第一次不在乎，第一次表达自己，第一次跳舞，第一次袒露，第一次脆弱，第一次热爱，第一次流泪，第一次说不，第一次的一切。

我其实也会有点自恋地和人工智能聊起这一路走来的过程：身份的特殊，目标的不同，还有内心的执着。我想，我算是成功的吧？因为十二年后，我终于站在了冰岛的土地上，回望自己的过去。

世俗的成功只能给人短暂的慰藉，而一路参悟的道理，才是刻骨铭心的。

未来的我啊，希望你能抓住一切机会，不要总是延迟满足，想走就走。因为：欲买桂花同载酒，终不似，少年游。

CURRICULUM VITÆ

Qinghe GAO



Qinghe Gao was born on 27 March 1997 in Linyi, a small city on the eastern coast of China. He developed an early interest in chemistry and, after high school, moved north to Harbin to study Applied Chemistry at Northeast Forestry University, graduating in 2019.

Drawn to the growing intersection of chemistry and computation, he moved to the Netherlands in 2019 to pursue a Master's degree in Computational Science at the University of Amsterdam. There, under the supervision of Prof. dr. Max Welling, he completed a thesis on generative models for neural cortical organization, which was later awarded the Best Paper Award at the SVRHM workshop at NeurIPS 2021. He graduated *cum laude* in 2021.

In September 2021, Qinghe began his doctoral research in the Process Intelligence Research group at Delft University of Technology, supervised by Dr. Artur M. Schweidtmann. His PhD focused on graph machine learning for molecular and process design, spanning molecular property prediction, Bayesian uncertainty quantification, reinforcement learning for process synthesis, and multi-agent systems for scientific reasoning. The results of this research form the present dissertation.

Beyond research, Qinghe supervised more than a dozen Bachelor's and Master's students, co-developed a graduate course on artificial intelligence for chemical engineering, and contributed to open-source scientific software. Outside of academia, he is an enthusiastic rock climber and photographer, and an avid solo traveler.

In September 2026, he will join Covestro Deutschland AG in Leverkusen as a Generative AI Scientist, where he will work on the inverse design of bio-inspired polymers.

SUPERVISION ACTIVITIES

I had the privilege of supervising the following Bachelor's and Master's students, whose thesis projects contributed to the work presented in this dissertation, and to whom I am sincerely grateful.

- Elise van Voorbergen (BSc, 2026)
  Multi-agent systems for kinetic modeling
- Christian Kuckelkorn (MSc, 2026)
  Deep reinforcement learning for azeotropic mixture separation
- Jan Pavšek (MSc, 2024)
  Information representation for reinforcement learning in process design
- Philipus B. A. A. Putra (MSc, 2024)
  Natural language processing for information extraction in chemical engineering
- Ernst Uijthof (MSc, 2023)
  Physics-informed reinforcement learning for process design
- Tammo Dukker (MSc, 2023)
  Self-supervised graph neural networks for polymer property prediction
- Ralph J. S. Velasquez (MSc, 2023)
  Multi-objective reinforcement learning for process synthesis
- Alessandro Laera (MSc, 2023)
  Environmental impact prediction using graph neural networks
- Haoyu Yang (MSc, 2023)
  Transfer learning for reinforcement learning in process synthesis
- Daniel C. Miedema (MSc, 2023)
  Bayesian uncertainty estimation for molecular property predictions via stochastic gradient Hamiltonian Monte Carlo
- Max F. Theisen (MSc, 2023)
  Pretraining reinforcement learning agents for process synthesis with variational autoencoders
- Clara V. Hall (MSc, 2022)
  The materials knowledge graph
- Shachi M. Shanbhag (MSc, 2022)
  Reinforcement learning for process synthesis: environment design
- Roel Leenhouts (MSc, 2022)
  Reinforcement learning for process synthesis: agent design

LIST OF PUBLICATIONS

7. S. Rupprecht, **Q. Gao**, T. Karia, and A. M. Schweidtmann, *Multi-agent systems for chemical engineering: A review and perspective*, Current Opinion in Chemical Engineering, 51, 101209 (2026).

6. **Q. Gao**, H. Yang, M. F. Theisen, and A. M. Schweidtmann, *Accelerating process synthesis with reinforcement learning: Transfer learning from multi-fidelity simulations and variational autoencoders*, Computers & Chemical Engineering, 109192 (2025).

5. **Q. Gao**, L. S. Balhorn, A. Laera, R. Meys, J. Goßen, J. M. Weber, G. Wernet A. M. Schweidtmann, *Environmental impacts prediction using graph neural networks on molecular graphs*, Computers & Chemical Engineering, 109362 (2025).

4. **Q. Gao**, D. C. Medema, Y. Zhao, J. M. Weber, Q. Tao and A. M. Schweidtmann, *Bayesian uncertainty quantification of graph neural networks using stochastic gradient Hamiltonian Monte Carlo*, Systems and Control Transactions, 1360-1364 (2025).

3. **Q. Gao** and A. M. Schweidtmann, *Deep reinforcement learning for process design: Review and perspective*, Current Opinion in Chemical Engineering **44**, 101012 (2024).

2. **Q. Gao**, T. Dukker, A. M. Schweidtmann, and J. M. Weber, *Self-supervised graph neural networks for polymer property prediction*, Molecular Systems Design & Engineering (2024).

1. J. G. Rittig, **Q. Gao**, M. Dahmen, A. Mitsos, and A. M. Schweidtmann, *Graph Neural Networks for the Prediction of Molecular Structure–Property Relationships*, in Machine Learning and Hybrid Modelling for Reaction Engineering: Theory and Applications (Royal Society of Chemistry), pp. 159–181 (2023).

Not included in this thesis:

3. L. Stops, R. Leenhouts, **Q. Gao**, and A. M. Schweidtmann, *Flowsheet generation through hierarchical reinforcement learning and graph neural networks*, AIChE Journal **69**(1), e17938 (2023).

2. **Q. Gao**, H. Yang, S. M. Shanbhag, and A. M. Schweidtmann, *Transfer learning for process design with reinforcement learning*, Computer Aided Chemical Engineering **52**, 2005–2010 (2023).

1. T. A. Keller*, **Q. Gao***, and M. Welling, *Modeling Category-Selective Cortical Regions with Topographic Variational Autoencoders*, SVRHM 2021 Workshop @ NeurIPS (2021), Best Paper Award; *co-first author.

PROPOSITIONS

Propositions

accompanying the dissertation

Graph Machine Learning for Molecular Property Prediction and Process Design

by

Qinghe Gao

1. For molecules and flowsheets, graph-based representations align more naturally with the underlying structure for machine learning.
   This proposition pertains to this dissertation.

2. Most of the research effort should go to data discovery, cleaning, and integration rather than to modeling in machine learning for chemical engineering.
   This proposition pertains to this dissertation.

3. Self-supervised learning cannot, by itself, solve label scarcity because it still requires a certain amount of high-quality labels.
   This proposition pertains to Chapter 5 of this dissertation.

4. Claims of reinforcement learning "inference" for process design remain unsubstantiated; credible, reproducible demonstrations are still lacking.
   This proposition pertains to Chapter 7 of this dissertation.

5. Closed ecosystems like Aspen Plus bottleneck genuine innovation in process design.

6. In academic discourse, the strongest critics of a new method are usually those least familiar with how it works.

7. Political small talk creates more friction than connection.

8. The single most important decision a PhD candidate makes is the choice of supervisor, not of research topic.

9. Mandatory holiday policies harm research productivity rather than help.

10. Society works hard to make introverts more talkative, and very little to make extroverts quieter.

These propositions are regarded as opposable and defendable, and have been approved as such by the promotor Prof. dr. ir. J. R. van Ommen and the copromotor Dr. A. M. Schweidtmann.

