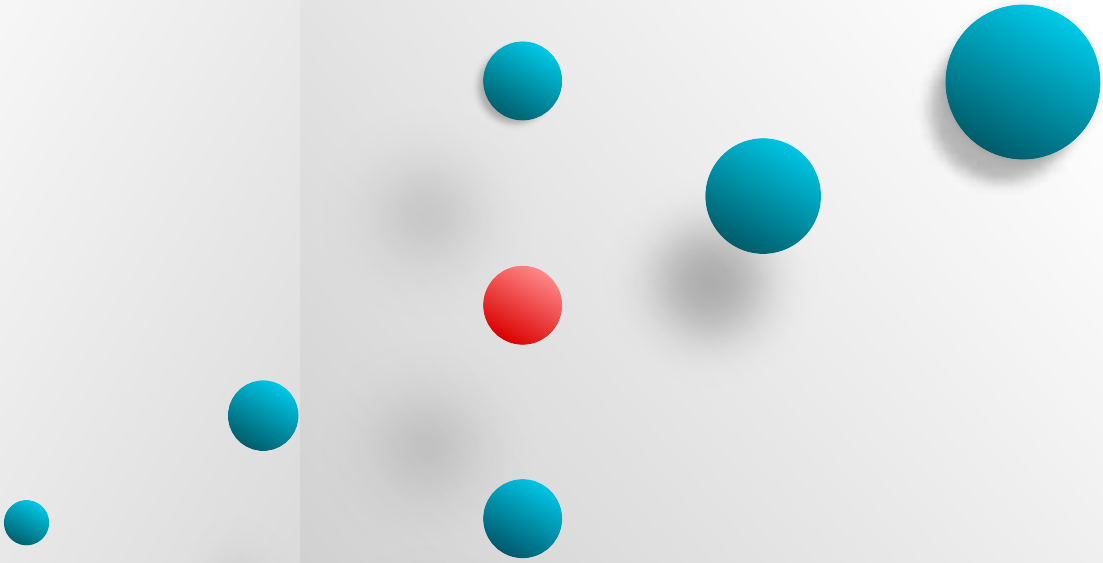


Learning with Spatial Biases in Vision Models

Robert-Jan Bruintjes

Propositions

accompanying the dissertation

Learning with Spatial Biases in Vision Models

by

Robert-Jan BRUINTJES

🎓 1. Inductive spatial biases in AI models trump spatial bias in data.

🎓 2. There is more scientific value in the results and methods of Deep Learning competitions than in scientific deep learning literature.

3. Improving vision models by improving data efficiency, rather than by adding more data, is the only way forward.

4. The better AI works, the less the public cares about how it works.

5. Although AI contributes to solving the climate crisis, it is also becoming a significant cause of climate change.

6. Lasting progress in deep learning can only come from a marriage of engineering and basic science.

7. Assessing PhD students in their collaborative skills will benefit their career.

8. Without government regulation, AI will not improve the lives of people on both sides of the wealth divide.

9. Stress induces searching for answers, rather than asking better questions.

10. Eurovision televoting will more accurately capture musical preference over political motivation if votes are limited to one vote per country per person.

🎓 Pertains to this dissertation.

These propositions are regarded as opposable and defendable,
and have been approved as such by the promotors
Prof. dr. ir. M.J.T. Reinders and Dr. J.C. van Gemert.

Learning with Spatial Biases in Vision Models



LEARNING WITH SPATIAL BIASES IN VISION MODELS

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
Wednesday 24 June 2026 at 12:30

by

Robert-Jan BRUINTJES

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof. dr. ir. M.J.T. Reinders,	Delft University of Technology, *promotor*
Dr. J.C. van Gemert,	Delft University of Technology, *promotor*
Independent members:	
Prof. dr. G.C.H.E. de Croon,	Delft University of Technology
Prof. dr. C.G.M. Snoek,	University of Amsterdam
Prof. dr. M. Loog,	Radboud University Nijmegen
Prof. dr. ir. R.L. Lagendijk,	Delft University of Technology
Dr. T.E.J. Mensink,	Google Research





Keywords: computer vision, spatial biases, visual priors, kernel size, scale, equivariance, data efficiency
Printed by: ProefschriftMaken.nl
Cover by: Robert-Jan Bruintjes

Copyright © 2026 by R. Bruintjes

ISBN 978-94-6518-343-5

An electronic copy of this dissertation is available at
https://repository.tudelft.nl/.

CONTENTS

Summary **ix**

Samenvatting **xi**

1 Introduction **1**
1.1 Inductive spatial biases . . . 3
1.2 Vision models . . . 5
1.3 Progress in inductive spatial biases . . . 9
1.4 Flexible inductive bias . . . 11
1.5 Organization of this thesis . . . 13
References . . . 17

2 Data-Efficient Challenges in Visual Inductive Priors: A Retrospective **21**
2.1 Introduction . . . 22
2.2 Related Works . . . 22
2.3 Challenges . . . 24
2.4 Submission and evaluation . . . 28
2.5 Methods . . . 29
2.6 Results . . . 34
2.7 Conclusion . . . 35
References . . . 37
Appendices . . . 43

3 What Affects Learned Equivariance in Deep Image Recognition Models? **63**
3.1 Introduction . . . 64
3.2 Related Works . . . 65
3.3 Method . . . 66
3.4 Experiments . . . 71
3.5 Conclusion . . . 78
References . . . 79

4 Learning to Adapt to Position Bias in Vision Transformer Classifiers **83**
4.1 Introduction . . . 84

4.2 Related work . . . 85
4.3 Position-SHAP . . . 88
4.4 Auto-PE . . . 89
4.5 Experiments . . . 90
4.6 Conclusion . . . 95
References . . . 97
Appendices . . . 101

5 Scale Learning in Scale-Equivariant Convolutional Networks 113
5.1 Introduction . . . 114
5.2 Related Work . . . 114
5.3 Method . . . 115
5.4 Experiments . . . 119
5.5 Discussion . . . 126
References . . . 129
Appendices . . . 131

6 FlexConv: Continuous Kernel Convolutions with Differentiable Kernel Sizes 133
6.1 Introduction . . . 134
6.2 Related Work . . . 137
6.3 Method . . . 138
6.4 Experiments . . . 142
6.5 Discussion . . . 148
6.6 Limitations . . . 149
6.7 Conclusion . . . 150
References . . . 151
Appendices . . . 155

7 Discussion 175
7.1 Position bias . . . 176
7.2 Scale bias . . . 177
7.3 Flexible inductive bias . . . 177
7.4 Methods for data-deficient settings . . . 178
7.5 Future outlook . . . 179
7.6 Final words . . . 180
References . . . 181

Acknowledgments 183

List of Publications **187**

Curriculum Vitæ **189**

SUMMARY

Much of the human experience is captured in the ever-increasing amounts of data that we humans generate. This data is used to build AI models that aid humans, e.g. by editing text, tracking traffic surrounding a self-driving car or detecting cancer in MRI scans. AI models typically get better when adding more training data. However, data cannot be sourced infinitely, and there are signs we are already running out of data to add to AI datasets. This is a concern, since AI models do not work well in settings where there is little data available (*data-deficient settings*), including important settings such as medical applications where gathering large datasets is costly and yields privacy concerns. We therefore desire *data-efficient* AI models: models that need less data, while still doing well.

This thesis investigates how design choices of neural network architectures (*inductive biases*) influence the data efficiency of vision models. In particular, we investigate the effects of modeling spatial biases apparent in training datasets. Such spatial biases are apparent through variability in *positions* and *scales* of imaged objects in the training data.

Vision models differ in their approach to modeling position information. Convolutional Neural Networks (CNNs) encode translation invariance through shared convolutional filters, enabling recognition of objects across positions with minimal data. However, this inductive bias removes positional information, which can hinder tasks requiring spatial reasoning. Vision Transformers (ViTs), in contrast, lack built-in spatial priors and rely on explicit positional embeddings to capture spatial structure, trading off data efficiency for global receptive fields.

Beyond position, scale variation presents a fundamental challenge: objects appearing at different sizes require models to generalize its learned recognition functions across scales. Conventional CNNs often learn separate representations for each scale, increasing data and compute demands. Scale-equivariant architectures address this by sharing representations across scales, and adaptive receptive field methods allow models to optimize kernel sizes for the observed distribution of object scales, improving robustness and efficiency.

To reconcile the challenges of modeling both types of spatial dataset biases, this thesis introduces flexible inductive biases: mechanisms that embed structural priors while allowing the model to modulate their influence based on dataset characteristics. Unlike fixed inductive biases, flexible inductive biases can be

activated when beneficial and suppressed when irrelevant, reducing the risk of negative transfer and improving data-efficiency.

Overall, the thesis demonstrates that spatial biases should be modelled as adaptive, dataset-dependent mechanisms rather than fixed architectural constraints. By systematically modeling position and scale biases in datasets and enabling their flexible use, this work provides a principled framework for designing data-efficient vision models that generalize robustly across diverse visual domains.

SAMENVATTING

Een groot deel van de menselijke ervaring is vastgelegd in de steeds grotere hoeveelheden data die wij mensen genereren. Deze data wordt gebruikt om AI-modellen te bouwen die mensen ondersteunen, bijvoorbeeld bij het bewerken van tekst, het in de gaten houden van omringend verkeer door een zelfrijdende auto of het detecteren van kanker in MRI-scans. Doorgaans worden AI-modellen beter wanneer er meer trainingsdata wordt toegevoegd. Echter, de hoeveelheid data in de wereld is niet oneindig, en er zijn aanwijzingen dat de beschikbare data voor AI-datasets al begint op te raken. Dit is zorgwekkend, aangezien AI-modellen slecht presteren in situaties waarin weinig data beschikbaar is (*data-deficiënte settings*), waaronder belangrijke toepassingen zoals medische diagnostiek , waar het verzamelen van grote datasets kostbaar is en privacyproblemen met zich meebrengt. We streven daarom naar *data-efficiënte* AI-modellen: modellen die minder data nodig hebben, maar toch goed presteren.

Dit proefschrift onderzoekt hoe ontwerpkeuzes van neurale netwerkarchitecturen (*inductieve biases*) de data-efficiëntie van visuele modellen beïnvloeden. In het bijzonder onderzoeken we de effecten van het modelleren van ruimtelijke biases die aanwezig zijn in trainingsdatasets. Zulke ruimtelijke biases uiten zich in variatie in de *posities* en *schalen* van geobserveerde objecten in de trainingsdata.

Visuele modellen verschillen in hoe ze positieinformatie modelleren. Convolutional Neural Networks (CNNs) zijn translatie-invariant door het gebruik van gedeelde convolutionele filters, waardoor objecten op verschillende posities kunnen worden herkend met minimale hoeveelheid data. Deze inductive bias verwijdert echter ook positie-informatie, wat nadelig kan zijn voor taken die ruimtelijk redeneren vereisen. Vision Transformers (ViTs) daarentegen hebben geen ingebouwde ruimtelijke priors en vertrouwen op expliciete positionele embeddings om ruimtelijke structuur vast te leggen, waarbij ze data-efficiëntie inruilen voor globale receptive fields.

Naast positie vormt schaalvariatie een fundamentele uitdaging: objecten die op verschillende groottes verschijnen vereisen dat modellen hun geleerde herkenningsfuncties over schalen heen generaliseren. Conventionele CNN's leren vaak afzonderlijke representaties voor elke schaal, wat de noodzaak aan data en rekenkracht vergroot. Schaal-equivariante architecturen pakken dit aan door representaties over schalen te delen, en methoden met adaptieve receptive fields

stellen modellen in staat de kernelgroottes te optimaliseren voor de verdeling van objectschalen in de data, wat de robuustheid en efficiëntie verbetert.

Om de uitdagingen van het modelleren van beide typen ruimtelijke datasetbiases te verzoenen, introduceert dit proefschrift *flexibele inductieve biases*: mechanismen die structurele priors inbedden, terwijl ze het model toestaan hun invloed te moduleren op basis van de kenmerken van de dataset. In tegenstelling tot vaste inductive biases kunnen flexibele inductive biases worden geactiveerd wanneer ze voordelig zijn en onderdrukt wanneer ze irrelevant zijn, waardoor het risico op negatieve transfer afneemt en de data-efficiëntie toeneemt.

Al met al toont dit proefschrift aan dat ruimtelijke biases het beste kunnen worden gemodelleerd als adaptieve, dataset-afhankelijke mechanismen in plaats van als vaste architecturale beperkingen. Door positie- en schaalbiases in datasets systematisch te modelleren en hun flexibele inzet mogelijk te maken, biedt dit werk een principieel kader voor het ontwerpen van data-efficiënte visuele modellen die robuust generaliseren over diverse visuele domeinen.

1

Introduction

1

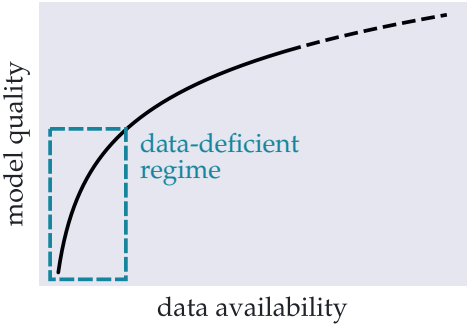
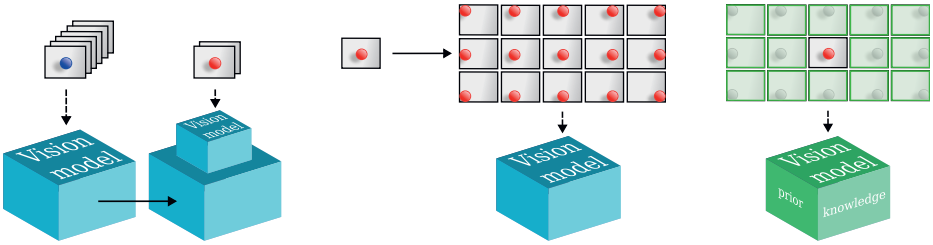


Figure 1.1: Scaling laws for Deep Learning [1–3], simplified. The more data is available to train Deep Learning models, the better they do. However, it is uncertain if we can keep adding more data to make better models. Also, many important applications (e.g. medical AI) live in the data-deficient regime: large amounts of data are not available, or are too expensive and/or privacy-sensitive to obtain.



(a) Transfer learning (b) Data augmentation (c) Inductive bias

Figure 1.2: Strategies for training data-efficient vision models. Dashed arrows representing training the model on the indicated data. (a) Transfer learning builds vision models on top of models trained on large similar datasets. (b) Data augmentation makes copies of data by applying transformations, e.g. shifting, selected based on prior knowledge about the task. (c) Using prior knowledge we can change the design of vision models (the *inductive bias*) to be more data-efficient.

In the course of our daily, ever-more online, lives, humanity creates vast and ever-increasing amounts of data. This data is used to build Artificial Intelligence (AI) models, which are *trained* on large datasets of examples labeled with the correct response: from each example in the training dataset, the model computes how wrong its response was, and corrects itself. AI models typically get better when adding more training data [1–3] (see Fig. 1.1).

However, data is not an infinite resource. While it may seem that every click, picture, or recording could be harnessed to train AI, high-quality labeled data is often costly and difficult to obtain. As it is, there are signs we are running out of data to add to AI datasets [4]. This is a concern, since AI models do not work well in settings where there is little data available (*data-deficient settings*), including important settings such as medical applications [5–7] where data collection involves expensive equipment, ethical considerations, or privacy restrictions. We therefore desire *data-efficient* AI models: models that need less data, while still doing well.

Researchers have developed several approaches to making AI models more data-efficient (see Fig. 1.2). In *transfer learning*, a model is first trained on a large dataset for a general task, such as recognizing objects in photographs. The knowledge it acquires—patterns of shapes, textures, and structures—can then be adapted to a new, often smaller dataset for a more specific task. However, transfer learning only works well if the data the large model was trained on, is similar to the data we train our model on [8]. *Data augmentation* is the process of artificially creating more data, by using prior knowledge to modify images. For example, by slightly shifting or mirroring the image one creates another image with the same desired response. A third approach, termed *inductive bias*, is to integrate prior knowledge about the data into the design of the vision model. According to the bias-variance tradeoff [9], the more inductive bias we use, the less data we need.

This thesis explores various techniques to improve applications of AI models in data-deficient settings. Chapter 2 shows that data augmentation is quite effective in data-deficient settings. Chapters 3 to 6 show that the prior knowledge used in both data augmentation and inductive bias can be combined to make AI models more data-efficient.

1.1 Inductive spatial biases

Vision models are AI models that work with images. Applications are plentiful: self-driving cars, automated radiology and crop monitoring are just a few examples. To power these applications, we train vision models on large datasets of

1

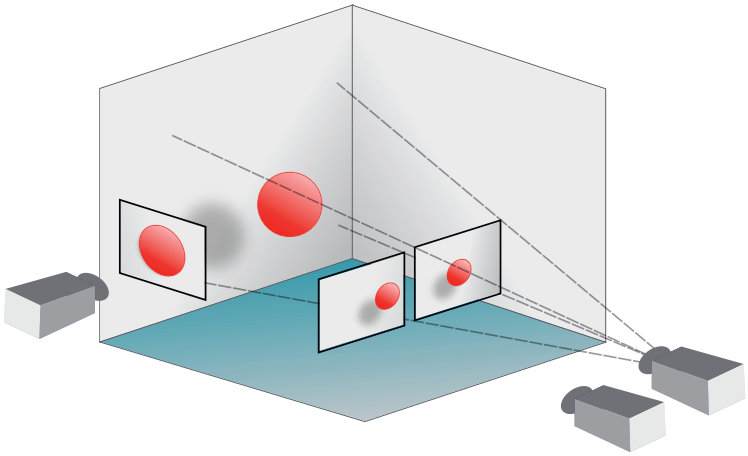


Figure 1.3: The image formation model describes the relationship between the position and size of objects in the 3D scene, and the position and scale of the objects in the 2D image. Regardless of the position of the camera, the captured image shows a red ball. Our AI model should be able to recognize the ball regardless of its position and scale in the image.



fish chicken shorts bird needle

Figure 1.4: Example images from the large dataset ImageNet [10] reflecting *photography bias*: the object of interest is framed in the center position of the picture, and its scale is chosen so that the object is entirely visible.

images. Fig. 1.3 shows how cameras create these 2D images from 3D scenes. The position of the camera determines the *position* and *scale* of objects in the image. We show an example scene where the task is to recognize the type of object ("ball"). A vision model should recognize the ball regardless of the position and scale of the object in the image. This is an example of prior knowledge that can be implemented as inductive bias in a vision model: by designing the vision model so that, no matter where or how large the red ball is, it gives the same response, we do not need to train this model anymore on images with the same ball in different positions or at different scales. This makes the model more data-efficient: less data is needed to train a model that works just as well.

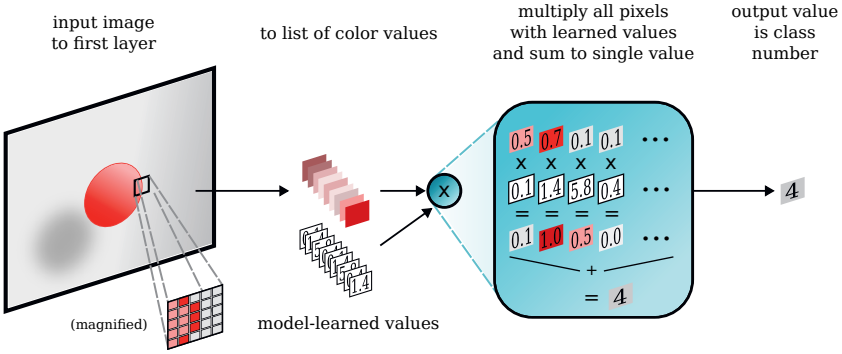
In the above example we assume that the position of the camera does not matter to recognize the ball, as long as the ball is in the image. However, what if certain camera positions happen much more than others, in a particular dataset? One example of such a *bias* is *photography bias*: photographers tend to put the object they are photographing in the center of the image [11, 12] (see Fig. 1.4). If vision models trained on such biased images use this bias, they can work better. For example, if there are multiple objects in the image, it is likely that the object to be recognized is the one in the center. This bias therefore is reflected in the *position* of the object in the image. Another example of bias may occur in the *scale* of the object in the image: photographers are likely to use the right amount of zoom to frame the object so that it fits the image. If the object is too large, it cannot be seen entirely; if it is too small, it is hard to see.

This thesis asks: do vision models learn to use such dataset biases? Or else, can we design them manually to use these biases to do better? This thesis is concerned with detecting and exploiting the presence of dataset biases in vision models. Specifically, we investigate *spatial dataset biases*: whether datasets contain bias on the *position* and *scale* of objects. We investigate whether vision models learn to use these spatial biases when they appear in the data, and how we can make vision models more data-efficient by integrating spatial dataset biases in their design as inductive biases.

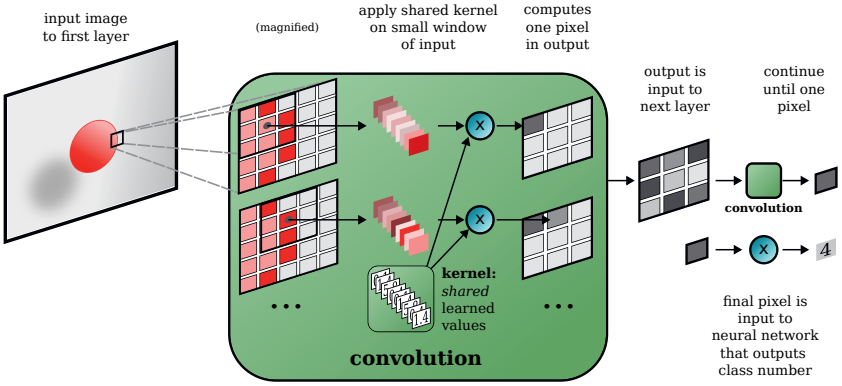
1.2 VISION MODELS

Vision models process an input image to compute a *response*. In the common task of image recognition, this response is a single number, which is the class number of the object in the image. In the popular benchmark dataset ImageNet [10], there are 1,000 classes, each assigned one class number called a *label* and each represented by roughly 1,000 images in the dataset. For example, the class "soccer ball" has label 805.

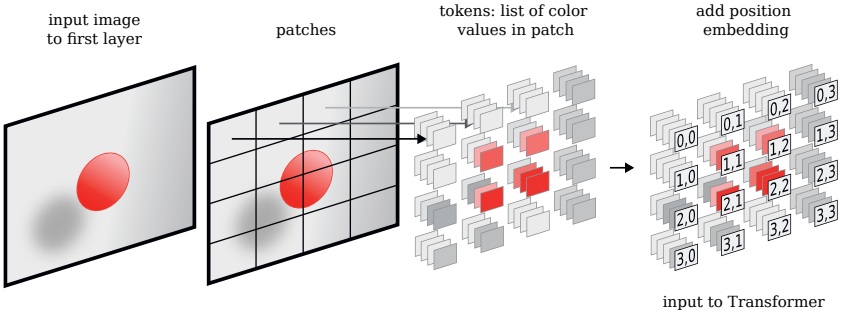
1



(a) Simplified one-layer neural network



(b) Simplified Convolutional Neural Network (CNN)



(c) Simplified Vision Transformer (ViT)

Figure 1.5: Vision models. (a) The basic neural network connects all pixels. (b) Each convolution in a CNN applies a shifted kernel over the input image, to compute a single pixel that is part of the input to the next layer. (c) ViT makes tokens out of image patches. Position embeddings are added, and the result is passed to a Transformer network.

When training a vision model, for each image in the training dataset, the vision model applies several learned functions to the pixels of the input image. These functions are simple mathematical functions, such as multiplication with specific learned values. The model then combines the outcomes of these functions to compute a single class number as response. If it computes the wrong response, the model learns by changing its functions a little bit to be more correct, before it moves on to the next image. Given enough images and corresponding class labels, as well as going through the dataset enough times, the model can learn to respond with the right class label in many images.

One basic implementation of a vision model is a *neural network*. It makes minimal assumptions about the structure of the image: it simply connects all input pixels to the response through multiple *layers* of learnable functions. Each layer of functions has as input the output of the previous layer. However, using a basic neural network as a vision model requires a lot of data to train a good model. This is because the model has to learn different functions for the same object appearing in different positions in the image. In practice, basic neural networks struggle to do well on vision tasks, such as image recognition, without access to enormous datasets and large amounts of computing power. Therefore, it is often necessary to infuse the neural network with design choices (i.e. inductive bias) based on our prior knowledge of the data. An *architecture* is such a particular neural network design. This thesis concerns two widely used architectures: the Convolutional Neural Network and the Vision Transformer. We investigate how they learn spatial biases, and how they can be changed to incorporate more inductive spatial biases.

Convolutional Neural Network. The *Convolutional Neural Network* (CNN) [13] is a neural network that consists mainly of layers of *convolution* operators. A convolution only connects pixels in a local neighborhood: it applies a moving *kernel* (a window) over the input image to compute values that are fed into some other operations, before going into the next convolution operator (see Fig. 1.5b). The kernel thereby computes a single function that is *shared* over all image positions. Because it is shared over all image positions, the inductive bias of the CNN architecture is based on the prior knowledge that the position of the camera in the scene is arbitrary and does not affect the object itself. This makes the CNN data-efficient [13]: instead of needing a hundred training images of the object in a hundred different locations, a CNN only needs to learn from one training sample to recognize the object in all locations.

The small "window" that the kernel creates on the input is called the *receptive field*, analogous to the human receptive field, which is the area of the environment

we can see in a glance. Vision models should have a large enough receptive field to "see" the entire image at once, so that they can recognize objects in the entire image. Basic neural networks have a global receptive field, since all inputs are connected to all outputs. CNNs on the other hand have a global receptive field if many convolutions with large enough kernel sizes are stacked in sequence. Configuring the CNN to have the right receptive field for the task at hand is therefore not trivial. In this thesis we propose a learnable kernel size method that allows a CNN to learn the optimal receptive field from the training data, removing the need for manual tuning of the receptive field (Chapter 6).

Vision Transformer. Transformers [14] are neural networks designed to work with language as input. Transformers were popularized initially in natural language processing, because they can efficiently process words in a sentence in parallel, which makes them more efficient than other language models. The *Vision Transformer* (ViT) [15] extends the Transformer to work with images. However, Transformers are difficult to train on images, because they lack inductive spatial bias: they connect all pixels together, like the basic neural network. The novelty of the Vision Transformer is to make patches of the image, then compress each patch to a single high-dimensional pixel called a *token* and feed the much smaller image into the Transformer, which connects all tokens (see Fig. 1.5c). Because there are many fewer tokens than pixels in the original image, this makes it feasible to connect all tokens. Notably, in contrast to the CNN, the internal operations in the Transformer are not able to distinguish the positions of tokens. Therefore, it is common practice to add to each token some information on its position, through a learnable *position embedding* [14]. This difference in processing of position information warrants comparing CNNs and ViTs when studying how position information is processed.

CNNs and ViTs are compared and contrasted in many recent works [16–22]. Most of note is that CNNs achieve strong performance even in relatively data-constrained regimes due to their inductive priors, whereas ViTs typically require substantially larger training datasets to achieve competitive results. Especially of importance to this thesis, is their differences in how they relate different parts of the image together. The CNN connects only pixels in a local neighborhood in each layer, while the ViT can relate objects far apart in the image in each layer. Also, while CNNs intrinsically encode image topology through their convolutional hierarchy, ViTs need positional embeddings to infer spatial relations. As such, these architectures model spatial dataset biases differently, and adding spatial inductive biases to them has to be done in a different way. Understanding how to integrate spatial inductive biases effectively in both CNNs and ViTs is

central to this thesis, as it directly impacts data-efficiency in vision tasks.

1.3 Progress in inductive spatial biases

In the following, we discuss this thesis' progress in integrating prior knowledge on dataset bias w.r.t. position and scale into vision models.

Modeling position bias. A central question in the design of vision architectures concerns the extent to which models should encode positional information. When an object in a 3D scene moves in parallel to the image plane, this is reflected by a *translation* (shift) of the object in the image. Classical convolutional architectures (CNNs) are built on the principle of translation *invariance*: they compute the same response regardless of where the object of interest appears in the image. This inductive bias makes the CNN data-efficient, because the model only has to learn a single recognition function to detect the object anywhere in the image. The counter-effect is that CNNs should not be able to know the position of objects in the image. This makes sense for tasks like image recognition, where the task is to assign a label to a single object in the image, regardless of where in the image is it. However, it may hinder models in other tasks where positional information is crucial, such as when localizing objects in the image. However, CNNs are accidentally *not* perfectly translation invariant. Factors such as boundary effects, striding, and pooling induce subtle positional dependencies [23, 24]. Where some works argue this is undesirable and should be fixed [23, 25], other works show that this unexpected ability may actually be help the model recognize objects [24, 26, 27].

In contrast to CNNs, ViTs *are* perfectly translation invariant, at least when the image is shifted by a multiple of patch size so that the patch edges line up. This means that ViTs cannot know the position of patches in the image. This limits the ViT's ability to reason about spatial layout. Interestingly, the original ViT paper [15] shows that, at least for ViTs, positions *should* be used by the model: when the model is allowed to know the position of patches through a position embedding, it does better, indicating that positional information is integral to visual recognition.

Given these seemingly conflicting findings, the question is: should vision models be able to know the position of objects of interest, or not? In Chapter 3 we show that some of the internal functions of vision models do not depend on the position of objects in the image. However, in Chapter 4 we show that whether the model uses the position of objects, depends on the spatial biases

apparent in the data: if the dataset has e.g. a photography bias, then letting the vision model know the position of objects helps the model do better. If however the position of the objects in the dataset is truly arbitrary, i.e. there is no dataset position bias, then prohibiting access to the position of objects helps the model do better. For example, a vision model trained on satellite imagery does worse when it learns to predict rivers always in the center of the image, as in fact they can appear anywhere in the image.

Taken together, these findings emphasize the importance of aligning inductive bias with the bias present in the training data. Rather than treating positional bias as universally good or bad, we find that the utility of positional information fundamentally depends on the data-generating process. By systematically analyzing how CNNs and ViTs encode, ignore, or exploit positional information, we provide new insights into the role of spatial inductive biases in achieving data-efficient learning.

Modeling scale bias. Another important dimension of spatial variation in images arises from changes in object scale. When the object of interest moves perpendicular to the image plane, i.e. closer or farther away from the camera, the scale of the object in the image becomes larger or smaller, respectively (see Fig. 1.3). Vision models can learn to recognize objects appearing at various scales in the image, but usually have to learn a different function for each scale, which means the model needs to see an example for each scale, making it data-inefficient.

In CNNs, scale variation is not encoded in the architecture. A convolutional filter detects patterns of a fixed spatial extent, determined by the kernel size. As a consequence, CNNs often learn separate filters for the same feature at different scales, which increases both the necessary parameter count and training dataset size. In practice, data augmentation strategies such as image resizing and cropping are commonly employed to expose models to multi-scale appearances. While effective, such approaches are data-inefficient, as they rely on repeated exposure to each scale.

To address this limitation, *scale equivariant* CNNs [28–30] extend the convolution operator to detect objects at multiple scales at once, sharing the same underlying recognition function. These architectures ensure that scaling an input object causes a predictable scaling of the corresponding network response. This inductive bias increases data-efficiency, since one scale-equivariant convolution only needs to learn to recognize an object at one scale, in order to recognize it at all other scales. However, the practical deployment of scale-equivariant CNNs introduces new challenges. Scale equivariant CNNs only work well if

their settings are tuned to the scale variance in the training data. Chapter 5 investigates how to learn automatically the right settings of scale equivariant models to match the distribution of scales in the training dataset.

This thesis also addresses the tuning of regular CNNs to scales apparent in the dataset. The convolution, the core operation of the CNN, has a predetermined kernel size. As discussed in Section 1.2, the kernel size determines the receptive field of the network, and therefore what scale of object can be detected. Therefore, using an appropriate kernel size is important to properly recognize objects appearing at particular scales. Chapter 6 proposes a method to automatically learn the kernel size during CNN training, allowing the model to optimize the receptive field to the scales in the data.

Taken together, these contributions investigate scale variation as a central dimension of inductive bias in vision models. By systematically analyzing how models encode, ignore, or adapt to scale transformations, this thesis contributes to a deeper understanding of the relationship between architectural design, dataset statistics, and data efficiency.

1.4 Flexible inductive bias

A central finding emerging from this thesis is that some datasets have less spatial biases than others. For example, in Chapter 4, we find that on some datasets an inductive position bias helps the model, while on others it does not. This variability highlights a fundamental limitation of fixed inductive biases: by hard-coding assumptions into the architecture, one risks introducing mismatches between model design and the biases in the data. To deal with this variance, we introduce the concept of *flexible inductive biases*: vision model operators with an inductive bias which can learn to model a dataset bias, but can also learn not to. Notably, these operators are designed to completely capture the learning of a particular spatial bias, using separate (and often less) parameters from the rest of the model parameters. This means the bias can be selectively learned or suppressed without interfering with the broader representational capacity of the model.

An example of a flexible inductive bias is the Auto-PE method of Chapter 4: it modulates the magnitude of the position embeddings in a Vision Transformer, making it directly and completely control how much dataset position bias a Vision Transformer learns to use. This way, Auto-PE can learn to turn off the position embeddings on datasets where using position bias does not help the model, while it retains the position embeddings on datasets where it helps. Even though ViT models without this flexible inductive bias can also learn to not use

Table 1.1: Contributions of each chapter as they relate to (inductive) spatial biases. Each chapter contributes methodology to either measure or add inductive bias in vision models (last three columns). We distinguish between inductive bias adding methods that fix the inductive bias in the method ("fixed inductive bias") and those that allow for (un)learning the particular bias ("flexible inductive bias"). These methodologies apply to either general spatial biases, position bias or scale bias specifically (first three columns).

	type of spatial bias			**measuring bias**	**adding inductive bias**	
	general	position	scale		fixed	flexible
Chapter 2: VIPriors	✓	-	-	-	✓	-
Chapter 3: Equivariance	-	✓	-	✓	-	-
Chapter 4: Position-SHAP	-	✓	-	✓	✓	✓
Chapter 5: Scale Learning	-	-	✓	-	✓	✓
Chapter 6: FlexConv	-	-	✓	-	-	✓

positional information, our models do better.

Chapters 5, 6 further extend this principle to scale-related biases. In Chapter 5, we show that flexible inductive bias mechanisms can be applied to match the empirical distribution of scales in the data. By parameterizing scale-equivariant operations so that they adapt to the biases in the data, the model achieves better alignment with real-world variability and demonstrates robustness to shifts in scale distributions not seen during training. In Chapter 6, we apply flexibility to the choice of receptive field, which is determined by choosing a fixed kernel size. We introduce a method for learnable kernel sizes, enabling receptive fields to be learned during training. Remarkably, the learned receptive fields converge to values consistent with decades of manually engineered design heuristics, but are obtained adaptively and with only a single training run.

From a conceptual perspective, flexible inductive biases occupy a middle ground between fixed inductive priors and purely data-driven learning. Fixed priors impose strong constraints that can reduce sample complexity but risk misalignment with data that does not contain the assumed bias. Fully unconstrained models, by contrast, avoid these risks but suffer from inefficiency, requiring large training datasets to learn even simple invariances. Flexible inductive biases reconcile these extremes: they embed structure into the architecture in a way

that facilitates efficient learning when the bias is present, yet allow the model to deactivate the bias when it is absent.

1.5 Organization of this thesis

The remainder of this thesis is composed of the following original contributions:

Chapter 2

- *Based on:* R. Bruintjes, A. Lengyel, O. S. Kayhan, D. Zambrano, N. Tömen, H. Jamali-Rad, and J. van Gemert. "Data-Efficient Challenges in Visual Inductive Priors: A Retrospective". In: *arXiv preprint arXiv:2506.08612* (2025)

- *Contribution of authors:*
  - R. Bruintjes: all aspects
  - A. Lengyel: workshop organization, technical implementation
  - O. S. Kayhan: workshop organization, technical implementation
  - D. Zambrano: workshop organization, technical implementation
  - N. Tömen: supervision and insights
  - H. Jamali-Rad: supervision and insights
  - J. C. van Gemert: workshop organization, supervision and insights

This chapter investigates which Deep Learning methods benefit training models in a data-deficient setting. Over the course of four years, we organized the "VIPriors: Visual Inductive Priors for Data-Efficient Deep Learning" workshop series, featuring four editions of data-impaired challenges. These challenges address the problem of training deep learning models for computer vision tasks with limited data. Participants are limited to training models from scratch using a low number of training samples and are not allowed to use any form of transfer learning. We aim to stimulate the development of novel approaches that incorporate prior knowledge to improve the data efficiency of deep learning models. Successful challenge entries make use of large model ensembles that mix Transformers and CNNs, as well as heavy data augmentation. Novel prior knowledge-based methods contribute to success in some entries.

1

Chapter 3

- *Based on:* R. Bruintjes, T. Motyka, and J. van Gemert. "What Affects Learned Equivariance in Deep Image Recognition Models?" In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2023, pp. 4839–4847
- *Contribution of authors:*
  - R. Bruintjes: technical implementation, supervision and insights
  - T. Motyka: technical implementation, insights
  - J. C. van Gemert: supervision and insights

This chapter investigates how vision models learn equivariant functions without inductive bias. Equivariance w.r.t. geometric transformations in neural networks improves data efficiency, parameter efficiency and robustness to out-of-domain perspective shifts. When equivariance is not designed into a neural network, the network can still learn equivariant functions from the data. We quantify this learned equivariance, by proposing an improved measure for equivariance. We find evidence for a correlation between learned translation equivariance and validation accuracy on ImageNet. We therefore investigate what can increase the learned equivariance in neural networks, and find that data augmentation, reduced model capacity and inductive bias in the form of convolutions induce higher learned equivariance in neural networks.

Chapter 4

- *Based on:* R. Bruintjes and J. van Gemert. "Learning to Adapt to Position Bias in Vision Transformer Classifiers". In: *arXiv preprint arXiv:2505.13137* (2025)
- *Contribution of authors:*
  - R. Bruintjes: all aspects
  - J. C. van Gemert: supervision and insights

This chapter investigates how Vision Transformers learn position bias. How discriminative position information is for image classification depends on the data. On the one hand, the camera position is arbitrary and objects can appear anywhere in the image, arguing for translation invariance. At the same time, position information is key for exploiting capture/center bias, and scene layout, *e.g.*: the sky is up. We show that *position bias*, the level to which a dataset is

more easily solved when positional information on input features is used, plays a crucial role in the performance of Vision Transformers image classifiers. To investigate, we propose **Position-SHAP**, a direct measure of position bias by extending SHAP to work with position embeddings. We show various levels of position bias in different datasets, and find that the optimal choice of position embedding depends on the position bias apparent in the dataset. We therefore propose **Auto-PE**, a single-parameter position embedding extension, which allows the position embedding to modulate its norm, enabling the unlearning of position information. Auto-PE combines with existing PEs to match or improve accuracy on classification datasets.

Chapter 5

- *Based on:* M. Basting, R. Bruintjes, T. Wiedemer, M. Kümmerer, M. Bethge, and J. van Gemert. "Scale Learning in Scale-Equivariant Convolutional Networks". In: *VISIGRAPP (2): VISAPP*. 2024

- *Contribution of authors:*
  - M. Basting: all aspects
  - R. Bruintjes: technical implementation, supervision and insights
  - J. C. van Gemert: supervision and insights

This chapter addresses scale bias. Objects can take up an arbitrary number of pixels in an image: Objects come in different sizes, and, photographs of these objects may be taken at various distances to the camera. These pixel size variations are problematic for CNNs, causing them to learn separate filters for scaled variants of the same objects which prevents learning across scales. This is addressed by scale-equivariant approaches that share features across a set of pre-determined fixed internal scales. These works, however, give little information about how to best choose the internal scales when the underlying distribution of sizes, or scale distribution, in the dataset, is unknown. In this work we investigate learning the internal scales distribution in scale-equivariant CNNs, allowing them to adapt to unknown data scale distributions. We show that our method can learn the internal scales on various data scale distributions and can adapt the internal scales in current scale-equivariant approaches.

Chapter 6

1

- *Based on:* D. W. Romero, R. Bruintjes, J. M. Tomczak, E. J. Bekkers, M. Hoogendoorn, and J. van Gemert. "FlexConv: Continuous Kernel Convolutions With Differentiable Kernel Sizes". In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: https://openreview.net/forum?id=3jooF27-0Wy
- *Contribution of authors:*
  - D.W. Romero: all aspects
  - R. Bruintjes: all aspects
  - J. M. Tomczak: supervision and insights
  - E. J. Bekkers: supervision and insights
  - M. Hoogendoorn: supervision and insights
  - J. C. van Gemert: supervision and insights

This chapter addresses the kernel size of convolutions. When designing Convolutional Neural Networks (CNNs), one must select the size of the convolutional kernels before training. Recent works show CNNs benefit from different kernel sizes at different layers, but exploring all possible combinations is unfeasible in practice. A more efficient approach is to learn the kernel size during training. However, existing works that learn the kernel size have a limited bandwidth. These approaches scale kernels by dilation, and thus the detail they can describe is limited. In this work, we propose *FlexConv*, a novel convolutional operation with which high bandwidth convolutional kernels of learnable kernel size can be learned at a fixed parameter cost. *FlexNets* model long-term dependencies without the use of pooling, achieve state-of-the-art performance on several sequential datasets, outperform recent works with learned kernel sizes, and are competitive with much deeper ResNets on image benchmark datasets. Additionally, FlexNets can be deployed at higher resolutions than those seen during training. To avoid aliasing, we propose a novel kernel parameterization with which the frequency of the kernels can be analytically controlled. Our novel kernel parameterization shows higher descriptive power and faster convergence speed than existing parameterizations. This leads to important improvements in classification accuracy.

Other publications Additional papers published during the research that are not integral to this thesis can be found in the List of publications on 187.

REFERENCES

[1] D. Hoiem, T. Gupta, Z. Li, and M. Shlapentokh-Rothman. "Learning Curves for Analysis of Deep Networks". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 4287–4296. URL: https://proceedings.mlr.press/v139/hoiem21a.html.

[2] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. "Scaling laws for neural language models". In: *arXiv preprint arXiv:2001.08361* (2020).

[3] T. Viering and M. Loog. "The shape of learning curves: a review". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.6 (2022), pp. 7799–7819.

[4] P. Villalobos, A. Ho, J. Sevilla, T. Besiroglu, L. Heim, and M. Hobbhahn. "Will we run out of data? Limits of LLM scaling based on human-generated data". In: *arXiv preprint arXiv:2211.04325* (2022).

[5] J. Chae and J. Kim. "An Investigation of Transfer Learning Approaches to Overcome Limited Labeled Data in Medical Image Analysis". In: *Applied Sciences* 13.15 (2023). ISSN: 2076-3417. DOI: 10.3390/app13158671. URL: https://www.mdpi.com/2076-3417/13/15/8671.

[6] A. M. Alsaleh, E. Albalawi, A. Algosaibi, S. S. Albakheet, and S. B. Khan. "Few-Shot Learning for Medical Image Segmentation Using 3D U-Net and Model-Agnostic Meta-Learning (MAML)". In: *Diagnostics* 14.12 (2024). ISSN: 2075-4418. DOI: 10.3390/diagnostics14121213. URL: https://www.mdpi.com/2075-4418/14/12/1213.

[7] Z. Hussain, F. Gimenez, D. Yi, and D. Rubin. "Differential data augmentation techniques for medical imaging classification tasks". In: *AMIA annual symposium proceedings*. Vol. 2017. 2018, p. 979.

[8] M. Raghu, C. Zhang, J. Kleinberg, and S. Bengio. "Transfusion: Understanding transfer learning for medical imaging". In: *Advances in neural information processing systems* 32 (2019).

[9] U. Von Luxburg and B. Schölkopf. "Statistical learning theory: Models, concepts, and results". In: *Handbook of the History of Logic*. Vol. 10. Elsevier, 2011, pp. 651–706.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[11] S. Fabbrizzi, S. Papadopoulos, E. Ntoutsi, and I. Kompatsiaris. "A survey on bias in visual datasets". In: *Computer Vision and Image Understanding* 223 (2022), p. 103552. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2022.103552. URL: https://www.sciencedirect.com/science/article/pii/S1077314222001308.

[12] A. Torralba and A. A. Efros. "Unbiased look at dataset bias". In: *CVPR 2011*. 2011, pp. 1521–1528. DOI: 10.1109/CVPR.2011.5995347.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. "Attention is all you need". In: *arXiv preprint arXiv:1706.03762* (2017).

[15] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929.

[16] W. Yu, M. Luo, P. Zhou, C. Si, Y. Zhou, X. Wang, J. Feng, and S. Yan. "Metaformer is actually what you need for vision". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10819–10829.

[17] J. Maurício, I. Domingues, and J. Bernardino. "Comparing Vision Transformers and Convolutional Neural Networks for Image Classification: A Literature Review". In: *Applied Sciences* 13.9 (2023). ISSN: 2076-3417. DOI: 10.3390/app13095521. URL: https://www.mdpi.com/2076-3417/13/9/5521.

[18] L. Deininger, B. Stimpel, A. Yuce, S. Abbasi-Sureshjani, S. Schönenberger, P. Ocampo, K. Korski, and F. Gaire. "A comparative study between vision transformers and CNNs in digital pathology". In: *arXiv preprint arXiv:2206.00389* (2022).

[19] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy. "Do vision transformers see like convolutional neural networks?" In: *Advances in neural information processing systems* 34 (2021), pp. 12116–12128.

[20] H. Yoo, L. Dai, S. Kim, and C.-B. Chae. "On the role of ViT and CNN in semantic communications: Analysis and prototype validation". In: *IEEE Access* 11 (2023), pp. 71528–71541.

[21] S. Cuenat and R. Couturier. "Convolutional neural network (cnn) vs vision transformer (vit) for digital holography". In: *2022 2nd International conference on computer, control and robotics (ICCCR)*. IEEE. 2022, pp. 235–240.

[22] E. Arkin, N. Yadikar, X. Xu, A. Aysa, and K. Ubul. "A survey: object detection methods from CNN to transformer". In: *Multimedia Tools and Applications* 82.14 (2023), pp. 21353–21383.

[23] O. S. Kayhan and J. C. v. Gemert. "On translation invariance in cnns: Convolutional layers can exploit absolute spatial location". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 14274–14285.

[24] M. A. Islam, M. Kowal, S. Jia, K. G. Derpanis, and N. D. Bruce. "Global pooling, more than meets the eye: Position information is encoded channel-wise in cnns". In: *Proceedings of the IEEE/CVF International conference on computer vision*. 2021, pp. 793–801.

[25] R. Zhang. "Making Convolutional Networks Shift-Invariant Again". In: *CoRR* abs/1904.11486 (2019). arXiv: 1904.11486. URL: http://arxiv.org/abs/1904.11486.

[26] M. A. Islam, S. Jia, and N. D. Bruce. "How much position information do convolutional neural networks encode?" In: *arXiv preprint arXiv:2001.08248* (2020).

[27] C. H. Lin, H.-Y. Tseng, H.-Y. Lee, M. K. Singh, and M.-H. Yang. "Unveiling The Mask of Position-Information Pattern Through the Mist of Image Features". In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, 23–29 Jul 2023, pp. 21204–21222. URL: https://proceedings.mlr.press/v202/lin23k.html.

[28] R. Ghosh and A. K. Gupta. "Scale steerable filters for locally scale-invariant convolutional neural networks". In: *arXiv preprint arXiv:1906.03861* (2019).

[29] D. Worrall and M. Welling. "Deep scale-spaces: Equivariance over scale". In: *Advances in Neural Information Processing Systems* 32 (2019).

[30] I. Sosnovik, M. Szmaja, and A. W. M. Smeulders. "Scale-Equivariant Steerable Networks". In: *CoRR* abs/1910.11093 (2019). arXiv: 1910.11093. URL: http://arxiv.org/abs/1910.11093.

[31] R. Bruintjes, A. Lengyel, O. S. Kayhan, D. Zambrano, N. Tömen, H. Jamali-Rad, and J. van Gemert. "Data-Efficient Challenges in Visual Inductive Priors: A Retrospective". In: *arXiv preprint arXiv:2506.08612* (2025).

[32] R. Bruintjes, T. Motyka, and J. van Gemert. "What Affects Learned Equivariance in Deep Image Recognition Models?" In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2023, pp. 4839–4847.

[33] R. Bruintjes and J. van Gemert. "Learning to Adapt to Position Bias in Vision Transformer Classifiers". In: *arXiv preprint arXiv:2505.13137* (2025).

[34] M. Basting, R. Bruintjes, T. Wiedemer, M. Kümmerer, M. Bethge, and J. van Gemert. "Scale Learning in Scale-Equivariant Convolutional Networks". In: *VISIGRAPP (2): VISAPP*. 2024.

[35] D. W. Romero, R. Bruintjes, J. M. Tomczak, E. J. Bekkers, M. Hoogendoorn, and J. van Gemert. "FlexConv: Continuous Kernel Convolutions With Differentiable Kernel Sizes". In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: https://openreview.net/forum?id=3jooF27-0Wy.

2

Data-Efficient Challenges in Visual Inductive Priors: A Retrospective

Deep Learning requires large amounts of data to train models that work well. In data-deficient settings, performance can be degraded. We investigate which Deep Learning methods benefit training models in a data-deficient setting, by organizing the "VIPriors: Visual Inductive Priors for Data-Efficient Deep Learning" workshop series, featuring four editions of data-impaired challenges. These challenges address the problem of training deep learning models for computer vision tasks with limited data. Participants are limited to training models from scratch using a low number of training samples and are not allowed to use any form of transfer learning. We aim to stimulate the development of novel approaches that incorporate prior knowledge to improve the data efficiency of deep learning models. Successful challenge entries make use of large model ensembles that mix Transformers and CNNs, as well as heavy data augmentation. Novel prior knowledge-based methods contribute to success in some entries.

This chapter has been published as:
R. Bruintjes, A. Lengyel, O. S. Kayhan, D. Zambrano, N. Tömen, H. Jamali-Rad, J. C. van Gemert. "Data-Efficient Challenges in Visual Inductive Priors: A Retrospective". In: *arXiv:2506.08612 [cs.CV], 2025.*

Code available at:
https://github.com/VIPriors/vipriors-challenges-toolkit

2.1 Introduction

Deep learning more and more depends on availability of large-scale training datasets. However, the cost of collecting and labeling such datasets scales with their size. Even if the issue of costly labeling can be avoided, training with large unlabeled datasets still uses large amounts of energy, contributing to carbon emissions [1, 2]. Furthermore, datasets and compute at such scale is limited to a few powerful big tech companies. Additionally, data at such a scale may not be available for some domains at all. The Visual Inductive Priors for Data-Efficient Deep Learning workshop (VIPriors) therefore encourages research in learning from small datasets, by way of combining the learning power of deep learning with hard-won prior knowledge from specific domains.

The VIPriors workshop ran for four editions at ICCV and ECCV from 2020 to 2023. Aside from featuring a paper track, each workshop hosts multiple challenges, where competitors train computer vision models on small training datasets, challenging them to find competitive solutions without the large quantities of data that power state-of-the-art deep computer vision models. The chosen tasks and datasets are tailored to be data-deficient and, where we are able to use custom datasets, contain prior knowledge that competitors can incorporate in their solutions. As far as we know, our VIPriors challenges are the only challenges that evaluate models in a data-deficient setting without allowing domain adaptation methods.

In this work, we describe all four editions of the VIPriors challenges [3–6]. Over the course of four years of challenges, we organize challenges on five distinct computer vision tasks: image classification, object detection, segmentation, action recognition and re-identification. For each task, we describe the challenge setup over the years, the accumulated results of all challenges and the most notable methods used by competitors.

We find that large model ensembles and heavy use of data augmentation are important factors in successful entries. Successful entries also mix Transformer-type architectures with CNN-type architectures in their ensembles. Even though a secondary aim of our challenge is for novel prior knowledge-based methods to be successful in a data-deficient setting, only a few such entries use prior knowledge to success.

2.2 Related Works

Related tasks. Our data-deficient setting is a supervised training setting with limited labeled samples available. There are other tasks that similarly aim to

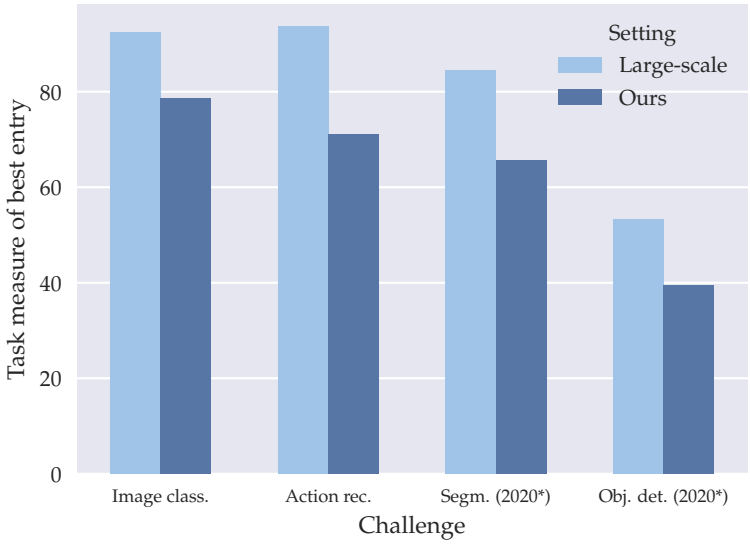


Figure 2.1: Comparing the winning entries in our challenges against those in corresponding large-scale benchmarks. Performance on our data-deficient settings are significantly worse, highlighting the need for research on data-deficient computer vision. For details, see Section 2.6.

enable deep learning with little to no data. The category of n-shot learning tasks is characterized by providing the learner with only n training samples, where n can be zero (zero-shot learning), one (one-shot learning) or a few samples (few-shot learning) [7, 8]. To solve this problem, learners often have to resolve to using additional information, for example by transfer learning from another training distribution.

In contrast, in our challenges we provide more data than used in few-shot learning, but much less than in common supervised training benchmarks. Where n-shot learning tasks often require alternative learning or transfer methods to perform, our data-deficient tasks can be approached with regular supervised learning, though performance may not be optimal. Our challenges explore the degree to which supervised learning struggles with this little amount of data and to which extent prior knowledge can alleviate these problems.

Related datasets. Several ImageNet-derivatives exist for data-deficient image classification. Mini-ImageNet [9] was proposed for evaluating few-shot methods

and is intended to be harder than CIFAR-10 but easier than full ImageNet. It contains 600 samples per class at full ImageNet image size. TinyImageNet [10] was created by the organizers of the CS231n course at Stanford University to serve as benchmark for a course project. It contains 500 samples per class, downsampled to a resolution of 64 pixels squared. Other unnamed subsets of ImageNet have also been proposed [11].

In a similar vein, we use subsampling of ImageNet to create small training datasets for our image classification challenge. In particular, we only use 50 samples per class, and in contrast to TinyImageNet we do not downsample the images.

As far as we know, our VIPriors challenges are the only challenges that evaluate models in a data-deficient setting without allowing domain adaptation methods.

2.3 CHALLENGES

Throughout the course of four editions of challenges, we have hosted five different computer vision tasks as challenges, with a varying number of tasks hosted each year. Common cause between all challenges is that the number of training samples are reduced to a small number. Where we adapt existing datasets we choose random subsets of the available data. In other cases, we adapt private datasets acquired through collaboration with industry, which are already so small to be appropriate for the challenge.

We choose to use small training datasets to realistically reproduce the setting of working in a data-deficient setting. Another goal of our challenges is to encourage the solutions to use visual prior knowledge of each task. To this end, we impose further rules to attempt to rule out usage of alternative methods that alleviate data scarcity. These rules are [6]:

- Models shall be trained from scratch with only the given dataset.
- The usage of other data rather than the provided training data, such as pre-training the models on other data and transfer learning methods, are prohibited. It is however allowed to train with synthetic data generated from the training data.

2.3.1 IMAGE CLASSIFICATION

Image classification is a cornerstone task in computer vision for its practical use and simple requirement of predicting only a single label. Architectures designed for many other tasks use image classification networks as backbones [12–15]. Innovations in image classification networks therefore have ripple effects throughout the whole field of computer vision. Therefore, we include this task in our challenges.

For the dataset we use a subset of ImageNet [16]. ImageNet has been the de facto benchmark standard for image classification since its inception [17]. Models trained on ImageNet, even with a reduced number of samples, transfer well to other tasks and distributions [18]. This motivates us to use ImageNet for our challenges, expecting that improvements in training models for a data-deficient ImageNet will translate to other tasks.

Our subset of ImageNet is a random sample of fifty images from 1,000 classes for training, validation and testing. This is more samples per class than for a typical few-shot setting, but much less than for a typical supervised learning setting. We evaluate models on top-1 accuracy.

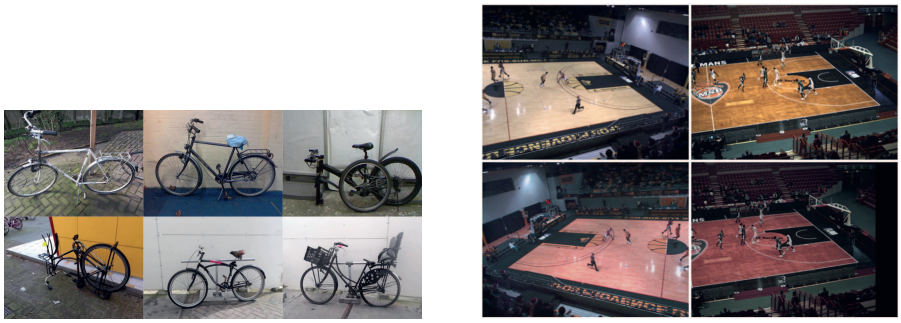
This challenge ran from 2020 until 2022. We did not run this challenge for the final edition in 2023, as we felt that competitor interest for challenges had shifted from image classification to more applied tasks like object detection and instance segmentation.

2.3.2 OBJECT DETECTION

Object detection is a widely applied task in computer vision. Its requirement to predict bounding boxes has driven researchers to invent widely varying architectures over the last decade, from ROI-based detectors [12, 13, 19] to YOLO-like dense predictors [20] and recently Transformers such as DETR [21]. For its widespread applied use, we include this task in our challenges.

We use the DelftBikes [22] dataset for our object detection challenge. The datasets contains 10,000 images of bikes, photographed from the side (Fig. 2.2a). Each image contains labels for 22 different bike parts that are annotated with bounding box, class and object state labels. We evaluate models using the same measure as used in the COCO benchmark [23]: Average Precision @ $0.50:0.95$ (AP).

This dataset is larger than other datasets used in our challenges, but still much smaller than other benchmark datasets for object detection such as COCO. We use this dataset for the strong consistency in bike pose and location of bike parts,



(a) DelftBikes (b) SynergySports Basketball

Figure 2.2: Example images from the a) DelftBikes and b) SynergySports Basketball dataset.

which enables competitors to use prior knowledge in their solutions.

This challenge ran from 2021 until 2023. For the 2020 edition, we ran the same challenge but used a subset of approximately 6,000 images from COCO [23].

2.3.3 Segmentation

Segmentation is a similar task to object detection, with the difference that instead of bounding boxes object instances are segmented by predicting pixel-wise masks. As masks can capture object outlines more precisely than bounding boxes, segmentation has become a popular benchmark in recent years [23]. We therefore include a segmentation task in our challenges.

Within the group of segmentation tasks, we distinguish semantic segmentation, where a single mask per category is predicted, and instance segmentation, where a mask is predicted for each object instance in the image. For our challenge, we use an instance segmentation dataset kindly provided by SynergySports[1], who collaborate with us on organizing this challenge. This dataset contains 18,232 still images taken from recordings of basketball games, where the objective is to segment basketball players and the ball. See Fig. 2.2b for example images. We evaluate models using Average Precision @ $0.50:0.95$ (AP).

We use this custom dataset for the consistency in camera pose and object appearances that enable competitors to use prior knowledge in their solutions.

This challenge ran from 2021 until 2023. For the 2020 edition, we instead

[1]https://synergysports.com

hosted a semantic segmentation task, using a subset of Cityscapes [24] called MiniCity. This subset consists of a train, validation and test set of 200, 100 and 200 images, respectively.

2.3.4 Action recognition

Video is the next frontier for computer vision. Action recognition is the task of classifying actions in video clips. Many action recognition are deep, heavy models that need to learn from a lot of data [25, 26]. We aim for our data-deficient action recognition challenge to further research into less data-hungry action recognition models.

To make a dataset for this challenge, we adapt the Kinetics400 dataset [27], the de facto benchmark for action recognition, into *Kinetics400VIPriors* by taking a subset. Our training set consists of approximately 100 clips per class (40,000 clips total), while the validation and test sets contain about 25 and 50 clips per class, respectively. For evaluation, we use the average classification accuracy across all classes on the test set.

This challenge ran in 2021 and 2022. For the 2020 edition, we instead used the UCF101 dataset [28], which in itself is already quite small. However, after the results of the 2020 challenge we felt that the high accuracies achieved showed that the competitors were not challenged enough with this dataset. After 2022, we chose to not run this challenge for the final edition, as we wanted to focus our efforts and this challenge did not receive as much interest from competitors as the other challenges.

2.3.5 Re-identification

Re-identification is the task of accurately retrieving the right person from an unseen gallery of photos given an unseen query photo. Typically, models learn to embed pictures of a training set of persons into vector embeddings to represent identities. At test time, the model is then presented with a query photo, which needs to match one of the unseen gallery images by matching the vector embedding. Having access to enough training identities is important. We therefore want to challenge competitors to achieve the same performance in a data-deficient setting.

We collaborated with SynergySports on creating a new, small dataset for this challenge. The provided dataset contains 954 identities and 18,232 images, taken from frame sequences of recordings of basketball games, similar to the data source for the Segmentation challenge (see Section 2.3.3 and Fig. 2.2b). The

dataset is split in training and testing identities. For the validation and test sets, each sequence is truncated to twenty frames, where the first frame is used as query image, and the other frames are used as gallery images. We use top-1 accuracy to evaluate models.

We use this custom dataset for the unique poses and appearances of the persons in it, namely players and referees in a basketball game. We expect that the specific and consistent appearance of this dataset enables competitors to use prior knowledge in their solutions.

This challenge only ran in 2021, where the winners of the challenge achieved a very high score. Because this showed that this setting was not challenging enough, we decided to not run this challenge again moving forward.

2.4 SUBMISSION AND EVALUATION

Our challenges are published on the VIPriors workshop website[2] and the CodaLab platform[3] [29], the latter of which also hosts the technical components of each competition. We make all training and validation data, as well as tooling and baseline models, available through CodaLab and a GitHub toolkit.[4]

To submit to the challenge, competitors are required to register with CodaLab, generate their models predictions over the test set on their own hardware, and upload these to CodaLab in a provided format. The CodaLab platform then automatically computes the evaluation measures and composed online rankings. The labels of the test set are withheld from competitors. As a measure to prevent overfitting to the test set, we limit the number of times an entry can be evaluated on the test set on CodaLab. We note some evidence suggesting that competitors evade these evaluation limits by registering with multiple accounts. As we cannot prove this, we do not address this.

At the time of each challenge closing, we compose official rankings by retaining only qualifying entries. To qualify, an entry has to be accompanied by a tech report, either published on the internet or submitted to the organizers by a given later date. We use these reports to verify the validity of the entry, and to enable our analysis of the methodology of each entry in the challenge reports. The final rankings are then published at the time of the live workshop program on the website and in the challenge report [3–6]. The full results over all editions of each

[2] https://vipriors.github.io

[3] 2020-2021: https://competitions.codalab.org/,
2022-2023: https://codalab.lisn.upsaclay.fr/

[4] https://github.com/VIPriors/vipriors-challenges-toolkit



Figure 2.3: Results of all challenges. Most challenges show improvement in the best submissions year by year. Hybrid architectures are prevalent under well-performing submissions. Each dot represents an entry in the final rankings. The 2020 edition of the object detection, segmentation and action recognition challenges used deprecated datasets with respect to the other years of the same challenge. We indicate what type of architectures were used as well as which entries used some novel methodology inspired by prior knowledge on the task.

challenge, including details on the methodology used, are given in the appendix (Tables 2..2 to 2..6). We visualize the results of all challenges in Fig. 2.3.

2.5 Methods

Analysis of the methods used by competitors shows patterns within individual challenges as well as patterns that generalize over all challenges. This section discusses these patterns and what we can conclude from them.

Model ensembling. Fig. 2.4 shows that many successful entries use large model ensembles. We speculate that the ease of use of model ensembling plays a role in competitors choosing ensembles: many architectures and backbones are available in various libraries, making it easy to plug in an extra model. Ensembles may also cover a wider range of inductive priors than a single model, making the ensemble more successful on a data-deficient task than a single model.

Architectures. As to the architectures used by competitors, our challenges experienced the advent of Transformers firsthand. Fig. 2.3 shows that Transformers start to see use in our challenges from 2021 onwards. We find that they are initially only used sporadically as backbones to a CNN-type architecture, but then increasingly as separate architectures, often combined with other inductive

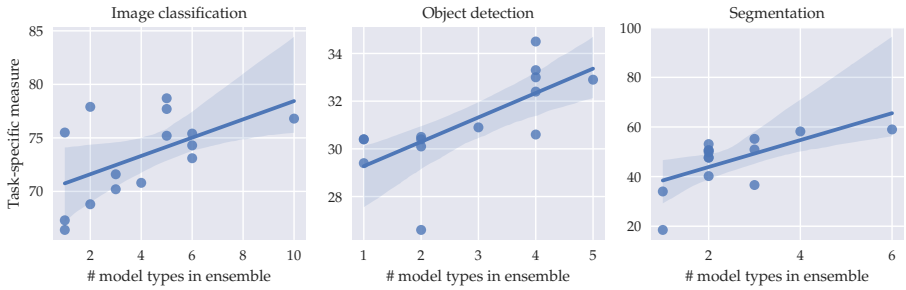


Figure 2.4: Results of selected challenges plotted against the number of distinct backbone types used in each entry. Each challenge shows a correlation where larger ensemble size correlates with better performance. Each dot represents an entry in the final rankings. Lines represent regression model fits performed by Seaborn [30]. We exclude the action recognition and re-identification challenges as they have too little entries to analyze.

priors such as CNN-type architectures in ensembles.

Data augmentation. Heavy use of data augmentation is apparent among successful entries, as shown in Fig. 2.5. The particular data augmentations differ per challenge, but the volume of augmentations is consistent. We expect data augmentations are successful in our challenges because they are an easy way to artificially increase the number of training samples in a way that uses prior knowledge on the task at hand. As for the popularity of data augmentations, we expect their ease of use may be a factor.

On prior knowledge in entries. Interestingly, we did not see as many novel prior-based methods being implemented in successful entries as we had expected. There are some cases where a novel method may have contributed to success, but only in combination with other factors like ensembling and data augmentation. We speculate that inventing a novel prior-based method is more strenuous on competitors, making them instead choose easier to implement methods such as ensembles and data augmentations.

Outside of novel methodological contributions, prior knowledge was used through pre-existing prior-based methods. One notable example is the use of BlurPool [31], which specifically enhances translation equivariance in CNNs. We posit that the extensive use of data augmentation policies can also be counted as a use of prior knowledge.



Figure 2.5: Results of selected challenges plotted against the number of data augmentations used in each entry. Each challenge shows a correlation where larger number of augmentation methods correlates with better performance. Each dot represents an entry in the final rankings. Lines represent regression model fits performed by Seaborn [30]. We exclude the action recognition and re-identification challenges as they have too little entries to analyze.

2.5.1 Image classification

In the entries of the image classification task, we note some specific trends. First of all, we confirm that the general trends of large model ensembles and heavy use of data augmentation hold for this task. Specifically, AutoAugment [32] and CutMix [33] are consistently used by almost all competitors. Transformers (specifically ConvNeXt [34]) appear in this task in 2022, and contribute to the top three entries in that year. Popular methods to boost performance include label smoothing and some form of extra training on hard or resized samples. Several entries use alternative or additional loss functions such as focal loss and ArcFace [35] loss. Almost all entries use regular supervised training, with a couple entries using some form of knowledge distillation without using additional data. No entries introduce any novel prior-based methodology. However, winning entries in 2021 and 2022 use BlurPool [31] to great success.

Winning teams and methods. Ma et al. from Xidian University are the winners of this challenge, scoring a top-1 accuracy of 78.7. They ensemble of five types of models, including both CNNs and Transformers. They apply various data augmentations to make the data more diverse. Additionally, they apply several optimization tricks, such as patching, hard fusion and random image cropping.

2.5.2 Object detection

In object detection, most entries use between two and four distinct backbone types in ensembles. Again, there is strong evidence for the benefit of large model ensembles. Many entries combine CNNs such as Cascade RCNN [14] or YOLO [20] with Transformers such as Swin [36] and ConvNeXt [34]. YOLO becomes the dominant architecture starting with version seven and eight of the YOLO framework [37, 38]. Correspondingly, the data augmentations implementations coupled with YOLOv8 are popular, as well as those implemented in the Albumentations library [39]. Specific augmentations used often are AutoAugment [32], Mosaic [40], MixUp [41] and some form of multi-scale augmentation. Ensembles are combined with Soft NMS [42], Weighted Boxes Fusion [43] or Model Soups [44] to combine predictions.

Overall, there are little entries that use novel prior-based methods. The exception is the 2023 edition, where the top two winning methods propose some novel prior-based methods.

Winning teams and methods. Zhao et al. from the Vision Intelligence Department of Meituan are the winners of this challenge, scoring an AP of 34.5. They use a Cascade RCNN [14] with a ConvNeXtV2 backbone [45]. They contribute a synthetic dataset created from horizontal and vertical recombinations of binary pairs of samples, on which they pre-train. Furthermore, they retrain the model on manually identified hard classes and apply SWA [46].

2.5.3 Segmentation

For segmentation, the same patterns apply when it comes to the number of models in ensembles and data augmentations. CutMix [33], MixUp [41], RandAugment [47], Moasic [40] and CopyPaste [48] are used by many entries, with CopyPaste being unique to this task. RCNN-type architectures (Cascade RCNN [14], Mask RCNN [13], then later on HTC [49]) dominate in this task. Where Transformers are used, they are almost exclusively used as backbone networks in an RCNN-type architecture. In the early editions of this challenge, backbones used are mostly HRNet [50] and CBSwin-T [51], while by 2023 there is a large variety of backbone networks, including anything from HTC, CBSwin, Mask2Former [52], DETR [21], ViT-Adapter [53] to Mask RCNN with FPN [54]. The best entries use (a variant of) Stochastic Weight Averaging [55] to ensemble multiple instances of the same model. Regular ensembling is also used through Weighted Boxes Fusion and Model Soups.

Interestingly, each edition of this challenge was won by an entry proposing a

novel prior-based method.

Winning teams and methods. Zhang et al. from Xidian University are the winners of this challenge, scoring an AP of 59.0. They propose a novel method called *Orthogonal Uncertainty Representation* (OUR), which broadens the geometric manifold [56] of underrepresented classes. Furthermore, they use multiple backbones with a Mask RCNN [13] model and a Seesaw loss [57].

2.5.4 Action recognition

For the task of action recognition, we have less entries to analyze. It is therefore hard to confirm the general trends on large model ensembles and heavy use of data augmentations for this task. On KineticsVIPriors, only ensembles are used, which combine two-stream methods with Transformer methods. Interestingly, the data augmentations used are the same as for appearance-only tasks (namely CutMix [33], MixUp [41], AutoAugment [32]). The only entry that makes video versions of these augmentations places last in the 2020 edition of this challenge. Outside of this entry, there are very little attempts to customize solutions to the priors of this task.

Winning teams and methods. Dave et al. from the University of Central Florida are the winners of this challenge with an average accuracy of 74%. They propose to combine several state-of-the-art methods that have shown promising results in data-deficient settings. They use convolutional (R3D [58] and I3D [25]) as well as attention-based (MViT [59]) models. Self-supervised pre-training (TCLR [60]) is applied to the convolutional methods, before they are fine-tuned with both RGB and optical flow frames. The Transformer-type model (MViT) is trained using only RGB frames.

2.5.5 Re-identification

As this challenge ran only for one year, with only four qualifying entries in the final ranking, we cannot draw any strong conclusions from this data. In fact, the entries are very similar in chosen methodology, which could indicate that the our sample of competitors is biased in some way. We do note that ensembles used only contain ResNets [61] and ResNet derivates [62, 63]. The most popular augmentation methods are Random Erasing [64], color jitter, random flipping and AutoAugment [32]. All entries use at least triplet loss [65] and circle loss [66]. Re-ranking [67] is used in the top three out of four entries. None of the entries use any form of novel prior-based methodology.

Table 2.1: Statistics and winners of each challenge.

Challenge	Entries	Qualified entries	Winning team	Winning score
Image classification	33	15	Tianzhi Ma, Zihan Gao, Wenxin He, Licheng Jiao *School of Artificial Intelligence, Xidian University.*	78.7
Object detection	87	17	Jiawei Zhao, Xuede Li, Xingyue Chen, Junfeng Luo. *Vision Intelligence Department (VID), Meituan.*	34.5
Segmentation	128	18	Junpei Zhang, Kexin Zhang, Rui Peng, Licheng Jiao, Fang Liu, Lingling Li, Yuting Yang. *Xidian University, Xi'an, Shaanxi.*	59.0
Action recognition	25	10	Ishan Dave, Naman Biyani, Brandon Clark, Rohit Gupta, Yogesh Rawat and Mubarak Shah. *Center for Research in Computer Vision (CRCV), University of Central Florida.*	74
Re-identification	12	4	Cen Liu, Yunbo Peng, Yue Lin. *NetEase Games AI Lab.*	96.5

Winning teams and methods. Liu et al. from NetEase Games AI Lab are the winners of this challenge with a top-1 accuracy of 96.5%. They use online difficult sample mining, in particular an algorithm similar to [68], so as to remove the hard, occluded annotations. They divide occluded annotations into partially and fully occluded annotations. Full occlusions are removed, while data augmentation is applied to the partial occlusions to create more of them. In this way, the robustness of the model is improved. Furthermore, they apply data augmentation using Local Grayscale Transform (LGT) and Random Erasing [64]. In particular, LGT ensures that color similarities in the jersey do not lead to problems. Furthermore, Liu et al. overrepresent IDs with less than twenty images to ensure balance in the training set. They use a model ensemble containing 24 model instances. Finally, common re-identification post-processing methods are applied: augmentation test, re-ranking [67] and query expansion [69].

2.6 RESULTS

Statistics on each challenge, as well as the overall winners, are given in Table 2.1. The segmentation and object detection challenges were the most popular challenges by some margin. We note that, except for in the 2022 edition of the action recognition challenge, the achieved results improve with every new edition, with a significant margin. The fact that state-of-the-art solutions improve so much in a year indicates that progress in computer vision still shows no signs of slowing down.

Comparison to large-scale settings. We compare the winning entries in our

challenges against the ranking leaders of comparable public large-scale benchmarks. For image classification and action recognition, our data-deficient settings are subsets of the same dataset used for a large-scale benchmark: ImageNet and Kinetics-400, respectively. For the object detection and segmentation challenges we used custom datasets from 2021 onward, while we used subsets of large-scale public datasets in 2020: COCO and Cityscapes, respectively. We therefore compare the winning entry in the 2020 edition of these challenges against the ranking leader in the corresponding challenge in 2020. The only edition of the re-identification challenge used a custom dataset and can therefore not be compared to any public large-scale dataset.

Fig. 2.1 shows the comparisons. We find gaps of around fifteen percentage points in all challenges. There are several possible explanations for these gaps. They may be due to our challenges having less competitors and therefore less competitive solutions. We do however expect that a large part of the gap is due to the difficulty of deep learning in a data-deficient setting without using additional data. We therefore posit that research into data-efficient deep learning is still quite necessary.

2.7 Conclusion

Over the course of four years, we organize the VIPriors challenges, challenging competitors to train computer vision models in a data-deficient setting. We provide more samples than in few-shot learning, but much less than given in large-scale benchmarks. Furthermore, networks had to be trained using only the provided data.

The goal of our challenges is to push progress in data-efficient deep learning, ideally by way of infusing prior knowledge into deep learning models. We can conclude that our challenges show progress in data-deficient deep learning, as winning entries improved in performance over the years. We also show that more research into data-efficient deep learning is necessary, as the winning entries in our data-deficient challenges score around 15 percentage points lower than their counterparts in large-scale benchmarks.

We analyze the methodology of challenge entries to make conclusions about successful methods for data-deficient deep learning. We find that winning entries make heavy use of model ensembling and data augmentation. Furthermore, mixing Transformers and CNNs is a recipe for success in our challenges.

As to our goal of encouraging the use of prior knowledge, we stimulate entries to use prior-based methods by introducing a jury prize for the best prior-based method per challenge. Whether or not the jury prize specifically contributes to

competitor's motivation, we can state that our challenges inspired some entries to use (novel) prior-based methods, and that some of these entries performed well in the challenges. Overall, however, methods not specifically designed for data-deficient settings, such as model ensembling and data augmentation, contributed most to success.

Limitations. Our challenges are specifically aimed at the setting of supervised training with little data. We rule out using any additional data, and therefore any transfer learning methods. An argument can be made that this is overly restrictive with respect to real-world data-deficient settings, where additional data may be available. A challenge that does not enforce this rule may be more representative of all possible approaches in a data-deficient setting.

Compared to large-scale benchmarks, the number of competitors in our challenges was relatively small. This may affect the confidence we have in general claims made based on the results of our challenges.

Finally, on the organizational side, there was some evidence that competitors evaded the evaluation limits on CodaLab by registering with multiple accounts. As we could not prove this, we did not address this.

Future work. Interestingly, we find that methods based on prior knowledge played only a small role in our challenges. We wonder if this means that the approach of integrating prior knowledge is truly not competitive with cheaply implemented methods such as model ensembling, or if there is another reason why these methods did not shine in our competition. Future work may investigate the efficacy of prior knowledge in deep learning more concretely.

Furthermore, future workshops may organize similar challenges around prior-based methods. We recommend devising an incentive through which prior-based methods are explicitly encouraged. This can be an organizational incentive such as our jury prize or an incentive integrated into the task measure, e.g. by scoring entries by some direct measure of their data efficiency [70].

With our challenges, we solely addressed limits on data availability. A similar concern is the energy cost of deep learning. Combining both concerns in a single challenge may change what methods are successful. For example, model ensembling is an effective way to improve performance with little effort but places a significant extra cost on the energy used by the model.

REFERENCES

[1] E. Strubell, A. Ganesh, and A. McCallum. "Energy and policy considerations for modern deep learning research". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 09. 2020, pp. 13693–13696.

[2] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni. "Green ai". In: *Communications of the ACM* 63.12 (2020), pp. 54–63.

[3] R. Bruintjes, A. Lengyel, M. Baptista-Ríos, O. S. Kayhan, and J. van Gemert. "VIPriors 1: Visual Inductive Priors for Data-Efficient Deep Learning Challenges". In: *arXiv preprint arXiv:2103.03768* (2021).

[4] A. Lengyel, R. Bruintjes, M. Baptista-Ríos, O. S. Kayhan, D. Zambrano, N. Tömen, and J. van Gemert. "Vipriors 2: visual inductive priors for data-efficient deep learning challenges". In: *arXiv preprint arXiv:2201.08625* (2022).

[5] R. Bruintjes, A. Lengyel, M. Baptista-Ríos, O. S. Kayhan, D. Zambrano, N. Tömen, and J. van Gemert. "Vipriors 3: Visual inductive priors for data-efficient deep learning challenges". In: *arXiv preprint arXiv:2305.19688* (2023).

[6] R. Bruintjes, A. Lengyel, M. Baptista-Ríos, O. S. Kayhan, D. Zambrano, N. Tömen, and J. van Gemert. "VIPriors 4: Visual Inductive Priors for Data-Efficient Deep Learning Challenges". In: *arXiv preprint arXiv:2406.18176* (2024).

[7] R. Kundu. *Everything you need to know about Few-Shot Learning*. en. https://blog.paperspace.com/few-shot-learning/. Accessed: 2024-07-09.

[8] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni. "Generalizing from a few examples: A survey on few-shot learning". In: *ACM computing surveys (csur)* 53.3 (2020), pp. 1–34.

[9] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.* "Matching networks for one shot learning". In: *Advances in neural information processing systems* 29 (2016).

[10] M. Ali. *Kaggle.com: Tiny ImageNet*. en. https://kaggle.com/competitions/tiny-imagenet. Accessed: 2024-07-09.

[11] L. Brigato, B. Barz, L. Iocchi, and J. Denzler. "Tune it or don't use it: Benchmarking data-efficient image classification". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 1071–1080.

[12] S. Ren, K. He, R. Girshick, and J. Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV].

[13] K. He, G. Gkioxari, P. Dollár, and R. Girshick. "Mask R-CNN". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322.

[14] Z. Cai and N. Vasconcelos. "Cascade r-cnn: Delving into high quality object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 6154–6162.

[15] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, C. C. Loy, and D. Lin. "Hybrid Task Cascade for Instance Segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[18] M. Huh, P. Agrawal, and A. A. Efros. "What makes ImageNet good for transfer learning?" In: *arXiv preprint arXiv:1608.08614* (2016).

[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[20] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[21] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. "End-to-end object detection with transformers". In: *European conference on computer vision*. Springer. 2020, pp. 213–229.

[22] O. S. Kayhan, B. Vredebregt, and J. C. van Gemert. "Hallucination In Object Detection–A Study In Visual Part Verification". In: *arXiv preprint arXiv:2106.02523* (2021).

[23] *COCO 2017 Object Detection Task*. https://cocodataset.org/. Accessed: 2024-07-10.

[24] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[25] J. Carreira and A. Zisserman. "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 4724–4733. DOI: 10.1109/CVPR.2017.502.

[26] C. Feichtenhofer, H. Fan, J. Malik, and K. He. "SlowFast Networks for Video Recognition". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 6201–6210. DOI: 10.1109/ICCV.2019.00630.

[27] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman. "The Kinetics Human Action Video Dataset". In: *arXiv preprint arXiv:1705.06950* (2017).

[28] K. Soomro, A. R. Zamir, and M. Shah. "UCF101: A dataset of 101 human actions classes from videos in the wild". In: *arXiv preprint arXiv:1212.0402* (2012).

[29] A. Pavao, I. Guyon, A.-C. Letournel, D.-T. Tran, X. Baro, H. J. Escalante, S. Escalera, T. Thomas, and Z. Xu. "CodaLab Competitions: An Open Source Platform to Organize Scientific Challenges". In: *Journal of Machine Learning Research* 24.198 (2023), pp. 1–6. URL: http://jmlr.org/papers/v24/21-1436.html.

[30] M. L. Waskom. "seaborn: statistical data visualization". In: *Journal of Open Source Software* 6.60 (2021), p. 3021. DOI: 10.21105/joss.03021. URL: https://doi.org/10.21105/joss.03021.

[31] R. Zhang. "Making Convolutional Networks Shift-Invariant Again". In: *ICML*. 2019.

[32] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. "Autoaugment: Learning augmentation policies from data". In: *arXiv preprint arXiv:1805.09501* (2018).

[33] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. "Cutmix: Regularization strategy to train strong classifiers with localizable features". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6023–6032.

[34] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. "A ConvNet for the 2020s". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022).

[35] J. Deng, J. Guo, N. Xue, and S. Zafeiriou. "Arcface: Additive angular margin loss for deep face recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4690–4699.

[36] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows". In: *International Conference on Computer Vision (ICCV)* (2021).

[37] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors". In: *arXiv preprint arXiv:2207.02696* (2022).

[38] *GitHub - ultralytics/ultralytics*. https://github.com/ultralytics/ultralytics. [Accessed 13-06-2024].

[39] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin. "Albumentations: Fast and Flexible Image Augmentations". In: *Information* 11.2 (Feb. 2020), p. 125. ISSN: 2078-2489. DOI: 10.3390/info11020125. URL: http://dx.doi.org/10.3390/info11020125.

[40] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: 2004.10934 [cs.CV].

[41] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. "mixup: Beyond Empirical Risk Minimization". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=r1Ddp1-Rb.

[42] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis. "Soft-NMS–improving object detection with one line of code". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5561–5569.

[43] W. W. Roman Solovyev and T. Gabruseva. "Weighted boxes fusion: Ensembling boxes from different object detection models". In: *Image and Vision Computing* 107 (2021), p. 104117.

[44] M. Wortsman, G. Ilharco, S. Y. Gadre, R. Roelofs, R. Gontijo-Lopes, A. S. Morcos, H. Namkoong, A. Farhadi, Y. Carmon, S. Kornblith, *et al.* "Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 23965–23998.

[45] S. Woo, S. Debnath, R. Hu, X. Chen, Z. Liu, I. S. Kweon, and S. Xie. "Convnext v2: Co-designing and scaling convnets with masked autoencoders". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 16133–16142.

[46] P. Izmailov, D. Podoprikhin, T. Garipov, D. P. Vetrov, and A. G. Wilson. "Averaging Weights Leads to Wider Optima and Better Generalization". In: *CoRR* abs/1803.05407 (2018). arXiv: 1803.05407. URL: http://arxiv.org/abs/1803.05407.

[47] E. D. Cubuk, B. Zoph, J. Shlens, and Q. Le. "RandAugment: Practical Automated Data Augmentation with a Reduced Search Space". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 18613–18624. URL: https://proceedings.neurips.cc/paper/2020/file/d85b63ef0ccb114d0a3bb7b7d808028f-Paper.pdf.

[48] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. D. Cubuk, Q. V. Le, and B. Zoph. "Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 2917–2927. DOI: 10.1109/CVPR46437.2021.00294.

[49] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, *et al.* "Hybrid task cascade for instance segmentation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4974–4983.

[50] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao. "Deep High-Resolution Representation Learning for Visual Recognition". In: *CoRR* abs/1908.07919 (2019). arXiv: 1908.07919. URL: http://arxiv.org/abs/1908.07919.

[51] T. Liang, X. Chu, Y. Liu, Y. Wang, Z. Tang, W. Chu, J. Chen, and H. Ling. "CBNetV2: A Composite Backbone Network Architecture for Object Detection". In: *arXiv preprint arXiv:2107.00420* (2021).

[52] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar. "Masked-attention mask transformer for universal image segmentation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 1290–1299.

[53] Z. Chen, Y. Duan, W. Wang, J. He, T. Lu, J. Dai, and Y. Qiao. "Vision transformer adapter for dense predictions". In: *arXiv preprint arXiv:2205.08534* (2022).

[54] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. "Feature pyramid networks for object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[55] H. Zhang, Y. Wang, F. Dayoub, and N. Sünderhauf. "SWA Object Detection". In: *arXiv preprint arXiv:2012.12645* (2020).

[56] Y. Ma, L. Jiao, F. Liu, S. Yang, X. Liu, and L. Li. "Curvature-balanced feature manifold learning for long-tailed classification". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 15824–15835.

[57] J. Wang, W. Zhang, Y. Zang, Y. Cao, J. Pang, T. Gong, K. Chen, Z. Liu, C. C. Loy, and D. Lin. "Seesaw Loss for Long-Tailed Instance Segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 9695–9704.

[58] K. Hara, H. Kataoka, and Y. Satoh. "Towards Good Practice for Action Recognition with Spatiotemporal 3D Convolutions". In: *2018 24th International Conference on Pattern Recognition (ICPR)*. 2018, pp. 2516–2521. DOI: 10.1109/ICPR.2018.8546325.

[59] H. Fan, B. Xiong, K. Mangalam, Y. Li, Z. Yan, J. Malik, and C. Feichtenhofer. "Multiscale Vision Transformers". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021.

[60] I. R. Dave, R. Gupta, M. N. Rizve, and M. Shah. "TCLR: Temporal Contrastive Learning for Video Representation". In: *arXiv preprint arXiv:2101.07974* (2021).

[61] K. He, X. Zhang, S. Ren, and J. Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[62] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, H. Lin, Z. Zhang, Y. Sun, T. He, J. Mueller, R. Manmatha, M. Li, and A. Smola. *ResNeSt: Split-Attention Networks*. 2020. arXiv: 2004.08955 [cs.CV].

[63] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. "Aggregated residual transformations for deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1492–1500.

[64] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang. "Random erasing data augmentation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 13001–13008.

[65] K. Q. Weinberger and L. K. Saul. "Distance metric learning for large margin nearest neighbor classification." In: *Journal of machine learning research* 10.2 (2009).

[66] Y. Sun, C. Cheng, Y. Zhang, C. Zhang, L. Zheng, Z. Wang, and Y. Wei. "Circle loss: A unified perspective of pair similarity optimization". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 6398–6407.

[67] Z. Zhong, L. Zheng, D. Cao, and S. Li. "Re-ranking person re-identification with k-reciprocal encoding". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1318–1327.

[68] A. Shrivastava, A. Gupta, and R. Girshick. "Training Region-Based Object Detectors With Online Hard Example Mining". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[69] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. "Total recall: Automatic query expansion with a generative feature model for object retrieval". In: *2007 IEEE 11th International Conference on Computer Vision*. IEEE. 2007, pp. 1–8.

[70] D. Hoiem, T. Gupta, Z. Li, and M. Shlapentokh-Rothman. "Learning curves for analysis of deep networks". In: *International conference on machine learning*. PMLR. 2021, pp. 4287–4296.

APPENDICES

The following pages contain the full rankings for all challenges.

Table 2..2: Overview of challenge entries for image classification challenge. J indicates jury prize. Bold-faced methods are contributions by the competitors. Due to confusion around the competition deadline in 2021, entries marked with † were awarded special rankings.

Image classification					
Rank	Team	Architectures, backbones	Data augmentation	Methods	Acc.
1 (2022)	**Ma et al.**	SE+PyramidNet, ResNeSt200e, ReXNet,EfficientNet-B8, ConvNeXt-XL* [1]	CutMix, AutoAugment [2], Stubborn Image Augmentation(SIA)	Label smoothing, AdvProp, Random image cropping and patching (RICP), extra training on stubborn images, hard fusion	**78.7**
2 (2022) & J (2022)	Lu et al.	HorNet, ConvNeXt [1]	Automix [3]	Cross-decoupled knowledge distillation [4], label smoothing	77.9
3 (2022)	Zuo et al.	CoAtNet, TResNet, Resnet50, Resnext50, EdgeNeXt	CutMix, Random erasing, MixUp, AutoAugment	Knowledge distillation between encoders	77.7
4 (2022)	Wang et al.	ResNeSt, TResNet, SE-ResNet, ReXNet, ECA-NFNet, ResNet-RS [5], Inception-ResNet, RegNet, EfficientNet, MixNet	AutoAugment, MixUp, CutMix, padding	label smoothing, train on larger images, data resampling	76.8
5 (2022)	She et al.	ResNeSt, Res2Net, Xception, DPN [6], EfficientNet, SENet	AutoAugment, MixUp	label smoothing, train on larger images, hard negative resampling	75.4
6 (2022)	Chen et al.	ResNeSt, EfficientNet, ReXNet, RegNetY	AutoAugment, MixUp, CutMix, ColorJitter	label smoothing, train on larger images, Exponential Moving Average on network parameters	70.8
1 (2021)	**Sun et al.**	ResNeSt [7]	AutoAugment [2], MixUp [8], CutMix [9]	BlurPool [10], stochastic depth [11]	**75.5**
2 (2021)	J. Wang et al.	ResNeSt [7], TResNet [12], RexNet [13], RegNet [14], Inception-ResNet [15]	AutoAugment [2], MixUp [8], CutMix [9]	label smoothing [16], DSB-Focalloss	75.2
2 (2021)†	Guo et al.	EfficientNet-b5/b6/b7 [17], DSK-ResNeXt101 [18] [19], ResNet-152 [20], SEResNet-152 [19]	AutoAugment [2], MixUp [8], CutMix [9], random erasing [21]	**Contrastive Regularization**, Mean Teacher [22], Symmetric Cross Entropy [23], label smoothing [16], dropout [24]	74.3
3 (2021) & J (2021)	T. Wang et al.	ResNeSt-101/200 [7], SEResNeXt-101 [19]	HorizontalFlip, FiveCrop, TenCrop, label smoothing [16]	**Iterative Partition-based Invariant Risk Minimization**	71.6
1 (2020)	**[25]**	EfficientNet-b5, EfficientNet-b6, ResNeSt-101, ResNest-200, DSK-ResNeXt50, DSK-ResNeXt101	Cutmix	**Dual Selective Kernel** based on SK [26] with anti-aliasing [10], center loss [27], tree supervision loss inspired by [28]	**73.1**
2 (2020)	[29]	ResNest-101 [7], TresNet-XL [12], SEResNeXt-101 [30]	AutoAugment [2], Cutmix [9]	combinations of cross-entropy loss and triplet loss [31] as well as ArcFace loss [32], label smoothing [16]	70.2
2 (2020)†	[33]	MOCO v2 [34] (teacher), ResNeXt101 [19] (student)	AutoAugment [2]	label smoothing [35], ten crops, model ensembling	68.8
3 (2020)	[36]	EfficientNet [17]	**Low Significant Bit swapping**, RandAugment [37]	**focal cosine loss** inspired by focal [38] and cosine [39] losses, Exponential Moving Average, dropout, drop connection, Plurality voting ensemble	67.3
4 (2020)	[40]	EfficientNet [17]	AutoAugment [2], MixUp [8]	model ensembling	66.4

Table 2..3: Overview of challenge entries for object detection challenge. J indicates jury prize. Bold-faced methods are contributions by the competitors. Due to confusion around the competition deadline in 2021, entries marked with † were awarded special rankings.

Object detection						
Rank	Team	Architectures, backbones	Data augmentation	Methods	AP (COCO)	AP (DelftBikes)
1 (2023)	**Zhao et al.**	Cascade RCNN [41], Swin T. [42], ConvNeXt [1], ConvNeXtV2 [43]	Albumentations [44], PhotoMetricDistortion, MixUp [8], Auto Augment V2 [2]	FPN [45], SWA [46], **recombined synthetic dataset**, **retraining hard classes**		**34.5**
2 (2023) & J (2023)	Lu et al.	Scaled-YOLOv4 [47], YOLOv7 [48], YOLOR [49], CBNetv2 [50]	Pre-training: random scaling, random flipping, color jitter; fine-tuning: Mosaic Augmentation [51], Copy-Paste [52], Mix-Up [8], Cutout [53]	Model Soups [54], Weighted Boxes Fusion [55], Test-Time Augmentation [56], **Image Uncertainty Weighted**		33.3
3 (2023)	Jing et al.	YOLOv7 [57], YOLOv8x-p2 [58], YOLOv8x, YOLOv8x-p6, Cascade RCNN [41]	Mosaic Augmentation [51], MixUp [8]	Weighted Boxes Fusion [55], Test-Time Augmentation [56], horizontal flip testing		30.6
4 (2023)	Wang et al.	YOLOv8 [58]	Mosaic Augmentation [51], MixUp [8]	SparK [59] pre-training, Weighted Boxes Fusion [55], Test-Time Augmentation [56], horizontal flip testing		30.4
5 (2023)	Sun et al.	YOLOv8 [58]	HSV, rotation, translation, scaling, shearing, flipping, Mosaic Augmentation [51], MixUp [8], Copy-Paste [52]	GAM [60], cross-validation		29.4
6 (2023)	*Team fha.ddd*	Faster RCNN [61], EfficientNet-V2 [62]	AutoAugment [2]			26.6
1 (2022)	**Lu et al.**	YOLOv4[51], YOLOv7[57], YOLOR [63], CBNetv2 [50]	Mosaic [51], MixUp [8], CopyPaste [52]	Weighted Boxes Fusion [55], TTA, Model Soups [54], Image Uncertainty Weighted		**33.0**
2 (2022)	Xu et al.	Cascade RCNN [41], Swin T. [42], ConvNext [1], ResNext [19], FPN [45]	AutoAugment [2], random flip, multi-scale augmentations [64]	MoCoV3 [65], MoBY [66], Soft-NMS [67], SSFPN[68], non-maximum weighted (NMW) [69]		32.9
3 (2022)	J. Zhao et al.	Cascade RCNN [41], Swin T. [42], Convnext [1], FPN [45]	Albu, MixUp [8], AutoAugment [2]	SWA [70], Hard classes retraining, Soft-NMS [67], pseudo labeling		32.4
4 (2022) & J (2022)	P. Zhao et al.	Cascade RCNN [41], Swin T. [42], Pyramid ViT [71]	Mosaic [51], MixUp [8]	SimMIM [72], GIOU loss [73], Soft-NMS [67]		30.9
1 (2021)	**Lu et al.**	YOLO 4-5 [51] [74]	Mosaic [51], MixUp [8], random color-jittering	Weighted Boxes Fusion [55]		**30.5**
1 (2021) †	**Zhang et al.**	Cascade RCNN [41], DCN [75]	Multi-scale augmentation	TTA, MoCo v2 [76], Soft-NMS [67], Class-specific IoU thresholds		30.4
2 (2021)	Niu et al.	Swin-T [42]	Hierarchical labeling	FPN [45], Soft-NMS [67], pseudo labeling		30.4
2 (2021) †	Luo et al.	Cascade RCNN [41], DCN [75]	Albu, Top-Bottom Cut	GCNet [77], SimSiam [78], Soft-NMS [67]		30.1
1 (2020)	**Shen et al.** [79]	Cascade-RCNN [41], Res2Net-152 [80], SeNet154 [30], ResNeSt-152 [7]	**Bbox-jitter**, GridMask [81], MixUp [8]	**infuse global context features**, Weighted Boxes Fusion [82]	**39.4**	
2 (2020)	Gu et al. [83]	Cascade R-CNN [41], ResNet-D [84], FPN [45], DCNv2 [85]	Albumentations [86], AutoAugment [87], Stitchers [88], **Mosaics-SC** based on Mosaic [51]	Libra R-CNN [89], Guided Anchoring [90], Generalized Attention [91], TSD [92], model ensembling	36.6	
3 (2020)	Luo et al. [93]	Scratch Mask R-CNN [94], ResNet-101 [20]	Albumentations [86]	Soft-NMS [67], extra weight on classification loss	35.1	

Table 2..4: Overview of challenge entries for segmentation challenge. J indicates jury prize. Bold-faced methods are contributions by the competitors.

Segmentation						
Rank	Team	Architectures, backbones	Data augmentation	Methods	mIoU (MiniCity)	mIoU (Basketball)
1 (2023)	**Zhang et al.**	HTC [95], Mask RCNN [96], Swin [42], ResNet [97], FPN [45], CBNet [98]	Geometric [99], color space, sharpness, noise injection, Copy-Paste [52]	**Orthogonal Uncertainty Representation**, Weighted Boxes Fusion [55], Seesaw loss [100], SWALP [101]		**59.0**
2 (2023)	Lu et al.	HTC [102], BEiTv2-L [103], ViT-Adapter [104], Internimage [105]	Mosaic Augmentation [51], Copy-Paste [52], Mix-Up [8], random brightness, random contrast, random saturation, random scale, random flip, sharpen and overlay, blur, Gaussian noise, grid-mask	GIoU loss [106], Soft NMS [67], **expert network** with SegFormer [107] and SeMask [108], Test-Time Augmentation [56], Model Soups [54], random scaling by Yunusov et al. [109]		58.2
3 (2023)	Liang et al.	Mask2Former [110], YOLOX [111], DETR [112]	Copy-Paste [52], rotation, mirroring, cropping, scaling, random brightness, random saturation, random contrast, random color equality, sharpness, random noise, random erasure, local erasure	Weighted Boxes Fusion [55], Seesaw loss [100], SWA [113], Soft NMS [67]		55.2
4 (2023) & J (2023)	Hsu et al.	HTC [102], Mask Scoring R-CNN [114], CB-SwinTransformer-Base [42, 98]	Players: RGB curve distortion; other objects: salt-and-pepper noise & brightness variations; GridMask [81]	**Basketball Court Detection**, GroupNorm [115], SWA [113], Model Soups [54]		50.9
1 (2022)	**[116]**	HTC [95], CBSwin-T [117]	**TS-DA**, **TS-IP** [116] Random scaling, cropping	CBFPN [117]		**53.1**
2 (2022)	Leng et al.	CBNetV2 [117], Swin Transformer-Large [42]	AutoAugment [2], ImgAug [118], Copy-Paste [119], Horizontal Flip and Multi-scale Training			50.6
2 (2022)	Lu et al.	HTC [95], CBSwin-T [117] ResNet [20] ConvNeXt [1] Swinv2 [42] CBNetv2 [117]	MixUp [8], Mosaic Task-Specific Copy-Paste [109] Color and geometric transformations	CBFPN [117] Group Normalization [115]		50.6
3 (2022)	Zhang et al.	HTC [95], CBSwin-T [117]	Location-aware MixUp, RandAugment [37], GridMask [81], Random scaling, CopyPaste [119], Multi-scale augmentation	Seesaw Loss [100] SWA [46], TTA		49.8
4 (2022)	Cheng et al.	HTC [95], CBSwin-T [117]	RandAugment Copy-Paste [119] GridMask	Mask Transfiner [120]		47.6
5 (2022) & J (2022)	[121]	ResNet [20]	Random flip and scale jitter	Sparse Instance Activation for Real-Time Instance Segmentation [121]		34.0
1 (2021) & J (2021)	**[109]**	HTC [95], CBSwin-T [117]	**Location-aware MixUp** [8], RandAugment [122], GridMask [81], Random scaling			**47.7**
2 (2021)	[123]	Cascade R-CNN [124], ResNet-101 [20]	Random brightness, color jitter, saturation, sharpening, blurring, noise, pixel shuffle, pixelization, filtering, hue transform	Switchable atrous convs. [125] Group normalization [115]		40.2
3 (2021)	Chen et al.	Cascade Mask-RCNN [41], SCNet [126], Swin [42]	HorizontalFlip, Random scale and crop			36.6
4 (2021)	Chen, Zheng	ResNet-50 [20]	Instaboost [127]	Seesaw Loss [100], Deformable Convolutions [75]		18.5
1 (2020)	**[128]**	HRNet [129]	**multi-scale** CutMix [9]		**65.64**	
2 (2020)	[130]	HRNetv2 [129]	Augmix [131]	Object Contextual Representations [132], Online Hard Example Mining [133], Frequency Weighted ensemble	65.61	
3 (2020)	[134]	HANet [135], ResNeSt [7]		**edge-preserving loss**, **pasting augmented crops of rare classes**	64.4	
4 (2020)	[136]	**EfficientSeg** using MobileNetV3 blocks [137]	random hue, random brightness, non-uniform scaling, random rotation, random flipping		58.0	
5 (2020)	[138]	UNet [139]	**CutMix Sprinkles** based on CutMix [9] and Progressive Sprinkles [140]		43.1	

Table 2..5: Overview of challenge entries for action recognition challenge. J indicates jury prize. Bold-faced methods are contributions by the competitors. Due to confusion around the competition deadline in 2021, entries marked with † were awarded special rankings.

Action recognition						
Rank	Team	Architectures, backbones	Data augmentation	Methods	Acc. (UCF101)	Acc. (KineticsViP)
1 (2022)	**Song et al.**	R(2+1)D [141], SlowFast [142], CSN [143], X3D [144], TANet [145], Timesformer [146]	Random flipping, TenCrop	Soft voting		**71**
2 (2022)	He et al.	SlowFast [142], Timesformer [146], TIN [147], TPN [148], X3D [144], Video Swin Transformers [149], R(2+1)D [141], DirecFormer [150].	AutoAugment [2], CutMix [9] random flip, grayscale, jitter, temporal aug., TenCrop, test-time aug.	Label smoothing		69
3 (2022) & J (2022)	Tan et al.	TSN [151], TANet [145], TPN [148], SlowFast [142], CSN [143], Video MAE [152]	MixUp [8], CutMix [9] MoCo [153], TVL-1 [154]		59	
1 (2021) & J (2021)	**Dave et al.**	R3D[155], I3D[156], MViT[157]		TCLR[158]		**74**
1 (2021) †	Wu et al.	TPN[148], Slowfast (slow path) [142]	MixUp [8], CutMix [9]			66
2 (2021)	Gao et al.	Swin [159], TPN [148], X3D[144], R2+1D[141], TimesFormer[146], Slowfast[142]				73
1 (2020)	**[160]**	I3D [156], C3D [161], R3D, R2+1D [141]	random crops, horizontal flipping, frame skipping		**90.8**	
2 (2020)	[162]	**modified** C3D [161]	corner cropping, horizontal flip	**Temporal Central Difference Convolution**, Rank Pooling [163]	88.3	
3 (2020)	[93]	SlowFast [142]	center/random crop, horizontal flip, normal/reverse video reproduction	TSM [164]	87.6	
4 (2020)	[165]	SlowFast-50 [142]	**RandAugment-T** based on RandAugment [37], **temporal extensions of** CutOut [53], MixUp [8], CutMix [9], random crop, random horizontal flip	model ensembling	86.0	

Table 2..6: Overview of challenge entries for re-identification challenge. J indicates jury prize. Bold-faced methods are contributions by the competitors.

Re-identification					
Rank	Team	Architectures, backbones	Data augmentation	Methods	Acc.
1 (2021)	**Liu et al.**	ResNet [20], ResNetSt [7], SE-ResNetXt [19] (24 models)	Difficult sample mining [133], Random Erasing [21], Local Grayscale Transform, affine transformations, pixel padding, random flip.	Triplet loss [166] and circle loss [167], with augmentation test, re-ranking [168], query expansion [169]	**96.5**
2 (2021) & J (2021)	Chen et al.	ResNet-IBN [20], SE ResNet-IBN [30] (5 models)	Video temporal mining, Random Erasing [21], pixel padding, random flip	Cross-entropy and triplet loss [166], with augmentation test, re-ranking [168], 6x schedule [170].	96.4
3 (2021)	Qi et al.	ResNet-IBN [20], 23OSNet on Stronger Baseline	Random Erasing [21], color jitter, random flip, AutoAugment [2]	Cross-entropy and triplet loss [166], re-ranking [168], query expansion [169]	94.2
4 (2021)	Zheng et al.	ResNet-IBN [20] w/ spatial and channel attention	Random Erasing and Patch [21], color jitter, random flip, AutoAugment [2]	Cross-entropy, triplet loss [166] and circle loss [167]	84.8

REFERENCES

[1] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. "A ConvNet for the 2020s". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022).

[2] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. "Autoaugment: Learning augmentation policies from data". In: *arXiv preprint arXiv:1805.09501* (2018).

[3] Z. Liu, S. Li, D. Wu, Z. Chen, L. Wu, J. Guo, and S. Z. Li. "Unveiling the Power of Mixup for Stronger Classifiers". In: *arXiv preprint arXiv:2103.13027* (2021).

[4] B. Zhao, Q. Cui, R. Song, Y. Qiu, and J. Liang. "Decoupled Knowledge Distillation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 11953–11962.

[5] I. Bello, W. Fedus, X. Du, E. D. Cubuk, A. Srinivas, T.-Y. Lin, J. Shlens, and B. Zoph. "Revisiting resnets: Improved training and scaling strategies". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 22614–22627.

[6] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng. "Dual path networks". In: *Advances in neural information processing systems* 30 (2017).

[7] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, H. Lin, Z. Zhang, Y. Sun, T. He, J. Mueller, R. Manmatha, M. Li, and A. Smola. *ResNeSt: Split-Attention Networks*. 2020. arXiv: 2004.08955 [cs.CV].

[8] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. "mixup: Beyond Empirical Risk Minimization". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=r1Ddp1-Rb.

[9] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. "Cutmix: Regularization strategy to train strong classifiers with localizable features". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6023–6032.

[10] R. Zhang. "Making Convolutional Networks Shift-Invariant Again". In: *ICML*. 2019.

[11] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. "Deep networks with stochastic depth". In: *European conference on computer vision*. Springer. 2016, pp. 646–661.

[12] T. Ridnik, H. Lawen, A. Noy, E. Ben Baruch, G. Sharir, and I. Friedman. "Tresnet: High performance gpu-dedicated architecture". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021, pp. 1400–1409.

[13] D. Han, S. Yun, B. Heo, and Y. Yoo. "Rethinking Channel Dimensions for Efficient Model Design". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 732–741.

[14] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár. "Designing network design spaces". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10428–10436.

[15] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. "Inception-v4, inception-resnet and the impact of residual connections on learning". In: *Thirty-first AAAI conference on artificial intelligence*. 2017.

[16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[17] M. Tan and Q. Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6105–6114.

[18] R. Bruintjes, A. Lengyel, M. Baptista-Ríos, O. S. Kayhan, and J. van Gemert. "VIPriors 1: Visual Inductive Priors for Data-Efficient Deep Learning Challenges". In: *arXiv preprint arXiv:2103.03768* (2021).

[19] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. "Aggregated residual transformations for deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1492–1500.

[20] K. He, X. Zhang, S. Ren, and J. Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[21] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang. "Random erasing data augmentation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 13001–13008.

[22] A. Tarvainen and H. Valpola. "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results". In: *arXiv preprint arXiv:1703.01780* (2017).

[23] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey. "Symmetric cross entropy for robust learning with noisy labels". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 322–330.

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[25] P. Sun, X. Jin, W. Su, Y. He, H. Xue, and Q. Lu. "A Visual Inductive Priors Framework for Data-Efficient Image Classification". In: *European Conference on Computer Vision Workshops*. Springer. 2020, pp. 511–520.

[26] X. Li, W. Wang, X. Hu, and J. Yang. "Selective kernel networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 510–519.

[27] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. "A discriminative feature learning approach for deep face recognition". In: *European conference on computer vision*. Springer. 2016, pp. 499–515.

[28] A. Wan, L. Dunlap, D. Ho, J. Yin, S. Lee, H. Jin, S. Petryk, S. A. Bargal, and J. E. Gonzalez. "NBDT: neural-backed decision trees". In: *arXiv preprint arXiv:2004.00221* (2020).

[29] Z. Luo, G. Li, and Z. Zhang. *A Technical Report for VIPriors Image Classification Challenge*. 2020. arXiv: 2007.08722 [cs.CV].

[30] J. Hu, L. Shen, and G. Sun. "Squeeze-and-excitation networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7132–7141.

[31] A. Hermans, L. Beyer, and B. Leibe. "In defense of the triplet loss for person re-identification". In: *arXiv preprint arXiv:1703.07737* (2017).

[32] J. Deng, J. Guo, N. Xue, and S. Zafeiriou. "Arcface: Additive angular margin loss for deep face recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4690–4699.

[33] B. Zhao and X. Wen. "Distilling visual priors from self-supervised learning". In: *European Conference on Computer Vision Workshops*. Springer. 2020, pp. 422–429.

[34] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. "Momentum Contrast for Unsupervised Visual Representation Learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[35] R. Müller, S. Kornblith, and G. Hinton. "When does label smoothing help?" In: *arXiv preprint arXiv:1906.02629* (2019).

[36] B. Kim, C. Kim, J. Lee, J. Song, and G. Park. *Data-Efficient Deep Learning Method for Image Classification Using Data Augmentation, Focal Cosine Loss, and Ensemble*. 2020. arXiv: 2007.07805 [cs.CV].

[37] E. D. Cubuk, B. Zoph, J. Shlens, and Q. Le. "RandAugment: Practical Automated Data Augmentation with a Reduced Search Space". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 18613–18624. URL: https://proceedings.neurips.cc/paper/2020/file/d85b63ef0ccb114d0a3bb7b7d808028f-Paper.pdf.

[38] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[39] B. Barz and J. Denzler. "Deep learning on small datasets without pre-training using cosine loss". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2020, pp. 1371–1380.

[40] M. E.-K. Qingfeng Liu Behnam Babagholami Mohamadabadi and J. Lee. "Diversification is All You Need : Towards Data Efficient Image Understanding". In: 2020. URL: https://openreview.net/forum?id=UPbbSsBzfEW.

[41] Z. Cai and N. Vasconcelos. "Cascade r-cnn: Delving into high quality object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 6154–6162.

[42] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows". In: *International Conference on Computer Vision (ICCV)* (2021).

[43] S. Woo, S. Debnath, R. Hu, X. Chen, Z. Liu, I. S. Kweon, and S. Xie. "Convnext v2: Co-designing and scaling convnets with masked autoencoders". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 16133–16142.

[44] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin. "Albumentations: Fast and Flexible Image Augmentations". In: *Information* 11.2 (2020). ISSN: 2078-2489. DOI: 10.3390/info11020125. URL: https://www.mdpi.com/2078-2489/11/2/125.

[45] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. "Feature pyramid networks for object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[46] P. Izmailov, D. Podoprikhin, T. Garipov, D. P. Vetrov, and A. G. Wilson. "Averaging Weights Leads to Wider Optima and Better Generalization". In: *CoRR* abs/1803.05407 (2018). arXiv: 1803.05407. URL: http://arxiv.org/abs/1803.05407.

[47] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. "Scaled-yolov4: Scaling cross stage partial network". In: *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*. 2021, pp. 13029–13038.

[48] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 7464–7475.

[49] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao. "You only learn one representation: Unified network for multiple tasks". In: *arXiv preprint arXiv:2105.04206* (2021).

[50] T. Liang, X. Chu, Y. Liu, Y. Wang, Z. Tang, W. Chu, J. Chen, and H. Ling. "Cbnetv2: A composite backbone network architecture for object detection". In: *arXiv preprint arXiv:2107.00420* (2021).

[51] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: 2004.10934 [cs.CV].

[52] M. Kisantal, Z. Wojna, J. Murawski, J. Naruniec, and K. Cho. "Augmentation for small object detection". In: *arXiv preprint arXiv:1902.07296* (2019).

[53] T. DeVries and G. W. Taylor. "Improved regularization of convolutional neural networks with cutout". In: *arXiv preprint arXiv:1708.04552* (2017).

[54] M. Wortsman, G. Ilharco, S. Y. Gadre, R. Roelofs, R. Gontijo-Lopes, A. S. Morcos, H. Namkoong, A. Farhadi, Y. Carmon, S. Kornblith, *et al.* "Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 23965–23998.

[55] W. W. Roman Solovyev and T. Gabruseva. "Weighted boxes fusion: Ensembling boxes from different object detection models". In: *Image and Vision Computing* 107 (2021), p. 104117.

[56] N. Moshkov, B. Mathe, A. Kertesz-Farkas, R. Hollandi, and P. Horvath. "Test-time augmentation for deep learning-based cell segmentation on microscopy images". In: *Scientific reports* 10.1 (2020), p. 5068.

[57] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors". In: *arXiv preprint arXiv:2207.02696* (2022).

[58] *GitHub - ultralytics/ultralytics*. https://github.com/ultralytics/ultralytics. [Accessed 13-06-2024].

[59] K. Tian, Y. Jiang, Q. Diao, C. Lin, L. Wang, and Z. Yuan. "Designing bert for convolutional networks: Sparse and hierarchical masked modeling". In: *arXiv preprint arXiv:2301.03580* (2023).

[60] S. Zhou, Y. Zhao, and D. Guo. "YOLOv5-GE Vehicle Detection Algorithm Integrating Global Attention Mechanism". In: *2022 3rd International Conference on Information Science, Parallel and Distributed Systems (ISPDS)* (2022), pp. 439–444. URL: https://api.semanticscholar.org/CorpusID:252112886.

[61] S. Ren, K. He, R. Girshick, and J. Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV].

[62] M. Tan and Q. Le. "Efficientnetv2: Smaller models and faster training". In: *International conference on machine learning*. PMLR. 2021, pp. 10096–10106.

[63] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao. "You only learn one representation: Unified network for multiple tasks". In: *arXiv preprint arXiv:2105.04206* (2021).

[64] W. K. Liu, S. Hao, T. Belytschko, S. Li, and C. T. Chang. "Multi-scale methods". In: *International Journal for Numerical Methods in Engineering* 47.7 (2000), pp. 1343–1361.

[65] X. Chen, S. Xie, and K. He. "An empirical study of training self-supervised vision transformers". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9640–9649.

[66] Z. Xie, Y. Lin, Z. Yao, Z. Zhang, Q. Dai, Y. Cao, and H. Hu. "Self-supervised learning with swin transformers". In: *arXiv preprint arXiv:2105.04553* (2021).

[67] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis. "Soft-NMS–improving object detection with one line of code". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5561–5569.

[68] M. Hong, S. Li, Y. Yang, F. Zhu, Q. Zhao, and L. Lu. "SSPNet: Scale selection pyramid network for tiny person detection from UAV images". In: *IEEE Geoscience and Remote Sensing Letters* 19 (2021), pp. 1–5.

[69] H. Zhou, Z. Li, C. Ning, and J. Tang. "Cad: Scale invariant framework for real-time object detection". In: *Proceedings of the IEEE international conference on computer vision workshops*. 2017, pp. 760–768.

[70] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson. "Averaging weights leads to wider optima and better generalization". In: *arXiv preprint arXiv:1803.05407* (2018).

[71] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao. "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 568–578.

[72] Z. Xie, Z. Zhang, Y. Cao, Y. Lin, J. Bao, Z. Yao, Q. Dai, and H. Hu. "Simmim: A simple framework for masked image modeling". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 9653–9663.

[73] G. I. O. Union. "A Metric and a Loss for Bounding Box Regression". In: *Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, S. Savarese//IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA*. 2019, pp. 658–666.

[74] G. Jocher and et. al. "YOLO v5". In: (2021). URL: https : / / github . com / %20ultralytics/yolov5.

[75] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. "Deformable convolutional networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 764–773.

[76] X. Chen, H. Fan, R. Girshick, and K. He. "Improved baselines with momentum contrastive learning". In: *arXiv preprint arXiv:2003.04297* (2020).

[77] Y. Cao, J. Xu, S. Lin, F. Wei, and H. Hu. "Gcnet: Non-local networks meet squeeze-excitation networks and beyond". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019, pp. 0–0.

[78] X. Chen and K. He. "Exploring simple siamese representation learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15750–15758.

[79] F. Shen, X. He, M. Wei, and Y. Xie. *A Competitive Method to VIPriors Object Detection Challenge*. Tech. rep. Huaqiao University, Wuhan University Of Technology, 2020.

[80] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. Torr. "Res2net: A new multi-scale backbone architecture". In: *IEEE transactions on pattern analysis and machine intelligence* 43.2 (2019), pp. 652–662.

[81] P. Chen, S. Liu, H. Zhao, and J. Jia. *GridMask Data Augmentation*. 2020. arXiv: 2001.04086 [cs.CV].

[82] R. Solovyev, W. Wang, and T. Gabruseva. "Weighted boxes fusion: Ensembling boxes from different object detection models". In: *Image and Vision Computing* (Feb. 2021), p. 104117. ISSN: 0262-8856. DOI: 10.1016/j.imavis.2021.104117. URL: http://dx.doi.org/10.1016/j.imavis.2021.104117.

[83] Y. Gu, Y. Pan, and S. Chen. *2nd Place Solution to ECCV 2020 VIPriors Object Detection Challenge*. 2020. arXiv: 2007.08849 [cs.CV].

[84] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li. *Bag of Tricks for Image Classification with Convolutional Neural Networks*. 2018. arXiv: 1812.01187 [cs.CV].

[85] X. Zhu, H. Hu, S. Lin, and J. Dai. "Deformable convnets v2: More deformable, better results". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 9308–9316.

[86] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin. "Albumentations: Fast and Flexible Image Augmentations". In: *Information* 11.2 (Feb. 2020), p. 125. ISSN: 2078-2489. DOI: 10.3390/info11020125. URL: http://dx.doi.org/10.3390/info11020125.

[87] B. Zoph, E. D. Cubuk, G. Ghiasi, T.-Y. Lin, J. Shlens, and Q. V. Le. *Learning Data Augmentation Strategies for Object Detection*. 2019. arXiv: 1906.11172 [cs.CV].

[88] Y. Chen, P. Zhang, Z. Li, Y. Li, X. Zhang, G. Meng, S. Xiang, J. Sun, and J. Jia. *Stitcher: Feedback-driven Data Provider for Object Detection*. 2020. arXiv: 2004.12432 [cs.CV].

[89] J. Pang, K. Chen, J. Shi, H. Feng, W. Ouyang, and D. Lin. *Libra R-CNN: Towards Balanced Learning for Object Detection*. 2019. arXiv: 1904.02701 [cs.CV].

[90] J. Wang, K. Chen, S. Yang, C. C. Loy, and D. Lin. *Region Proposal by Guided Anchoring*. 2019. arXiv: 1901.03278 [cs.CV].

[91] X. Zhu, D. Cheng, Z. Zhang, S. Lin, and J. Dai. *An Empirical Study of Spatial Attention Mechanisms in Deep Networks*. 2019. arXiv: 1904.05873 [cs.CV].

[92] G. Song, Y. Liu, and X. Wang. *Revisiting the Sibling Head in Object Detector*. 2020. arXiv: 2003.07540 [cs.CV].

[93] Z. Luo and L. Che. *VIPriors Object Detection Challenge*. 2020. arXiv: 2007.08170 [cs.CV].

[94] R. Zhu, S. Zhang, X. Wang, L. Wen, H. Shi, L. Bo, and T. Mei. "ScratchDet: Training single-shot object detectors from scratch". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2268–2277.

[95] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, C. C. Loy, and D. Lin. "Hybrid Task Cascade for Instance Segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[96] K. He, G. Gkioxari, P. Dollár, and R. Girshick. "Mask R-CNN". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322.

[97] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[98] Y. Liu, Y. Wang, S. Wang, T. Liang, Q. Zhao, Z. Tang, and H. Ling. "Cbnet: A novel composite backbone network architecture for object detection". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 07. 2020, pp. 11653–11660.

[99] M. Paschali, W. Simson, A. G. Roy, R. Göbl, C. Wachinger, and N. Navab. "Manifold exploring data augmentation with geometric transformations for increased performance and robustness". In: *Information Processing in Medical Imaging: 26th International Conference, IPMI 2019, Hong Kong, China, June 2–7, 2019, Proceedings 26*. Springer. 2019, pp. 517–529.

[100] J. Wang, W. Zhang, Y. Zang, Y. Cao, J. Pang, T. Gong, K. Chen, Z. Liu, C. C. Loy, and D. Lin. "Seesaw Loss for Long-Tailed Instance Segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 9695–9704.

[101] G. Yang, T. Zhang, P. Kirichenko, J. Bai, A. G. Wilson, and C. De Sa. "SWALP: Stochastic weight averaging in low precision training". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 7015–7024.

[102] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, *et al.* "Hybrid task cascade for instance segmentation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4974–4983.

[103] Z. Peng, L. Dong, H. Bao, Q. Ye, and F. Wei. "A unified view of masked image modeling". In: *arXiv preprint arXiv:2210.10615* (2022).

[104] Z. Chen, Y. Duan, W. Wang, J. He, T. Lu, J. Dai, and Y. Qiao. "Vision transformer adapter for dense predictions". In: *arXiv preprint arXiv:2205.08534* (2022).

[105] W. Wang, J. Dai, Z. Chen, Z. Huang, Z. Li, X. Zhu, X. Hu, T. Lu, L. Lu, H. Li, *et al.* "Internimage: Exploring large-scale vision foundation models with deformable convolutions". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 14408–14419.

[106] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. "Generalized intersection over union: A metric and a loss for bounding box regression". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 658–666.

[107] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo. "SegFormer: Simple and efficient design for semantic segmentation with transformers". In: *Advances in neural information processing systems* 34 (2021), pp. 12077–12090.

[108] J. Jain, A. Singh, N. Orlov, Z. Huang, J. Li, S. Walton, and H. Shi. "Semask: Semantically masked transformers for semantic segmentation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 752–761.

[109] J. Yunusov, S. Rakhmatov, A. Namozov, A. Gaybulayev, and T.-H. Kim. *Instance Segmentation Challenge Track Technical Report, VIPriors Workshop at ICCV 2021: Task-Specific Copy-Paste Data Augmentation Method for Instance Segmentation*. 2021. arXiv: 2110.00470 [cs.CV].

[110] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar. "Masked-attention mask transformer for universal image segmentation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 1290–1299.

[111] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun. "Yolox: Exceeding yolo series in 2021". In: *arXiv preprint arXiv:2107.08430* (2021).

[112] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. "End-to-end object detection with transformers". In: *European conference on computer vision*. Springer. 2020, pp. 213–229.

[113] H. Zhang, Y. Wang, F. Dayoub, and N. Sünderhauf. "SWA Object Detection". In: *arXiv preprint arXiv:2012.12645* (2020).

[114] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. Wang. "Mask Scoring R-CNN". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 6402–6411. DOI: 10.1109/CVPR.2019.00657.

[115] Y. Wu and K. He. "Group Normalization". In: *ECCV*. 2018.

[116] B. Yan, X. Zhao, Y. Li, and H. Wang. "Task-Specific Data Augmentation and Inference Processing for VIPriors Instance Segmentation Challenge". In: (2022). URL: https://arxiv.org/abs/2211.11282.

[117] T. Liang, X. Chu, Y. Liu, Y. Wang, Z. Tang, W. Chu, J. Chen, and H. Ling. "CBNetV2: A Composite Backbone Network Architecture for Object Detection". In: *arXiv preprint arXiv:2107.00420* (2021).

[118] A. B. Jung. *imgaug*. https://github.com/aleju/imgaug. [Online; accessed 30-Oct-2018]. 2018.

[119] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. D. Cubuk, Q. V. Le, and B. Zoph. "Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 2917–2927. DOI: 10.1109/CVPR46437.2021.00294.

[120] L. Ke, M. Danelljan, X. Li, Y.-W. Tai, C.-K. Tang, and F. Yu. "Mask Transfiner for High-Quality Instance Segmentation". In: *CVPR*. 2022.

[121] T. Cheng, X. Wang, S. Chen, W. Zhang, Q. Zhang, C. Huang, Z. Zhang, and W. Liu. "Sparse Instance Activation for Real-Time Instance Segmentation". In: *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*. 2022.

[122] E. D. Cubuk, B. Zoph, J. Shlens, and Q. Le. "RandAugment: Practical Automated Data Augmentation with a Reduced Search Space". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 18613–18624. URL: https://proceedings.neurips.cc/paper/2020/file/d85b63ef0ccb114d0a3bb7b7d808028f-Paper.pdf.

[123] B. Yan, F. Qi, L. Cao, and H. Wang. *The Second Place Solution for ICCV2021 VIPriors Instance Segmentation Challenge*. 2021. arXiv: 2112.01072 [cs.CV].

[124] Z. Cai and N. Vasconcelos. "Cascade R-CNN: Delving Into High Quality Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[125] S. Qiao, L.-C. Chen, and A. Yuille. *DetectoRS: Detecting Objects with Recursive Feature Pyramid and Switchable Atrous Convolution*. 2020. arXiv: 2006.02334 [cs.CV].

[126] T. Vu, K. Haeyong, and C. D. Yoo. "SCNet: Training Inference Sample Consistency for Instance Segmentation". In: *AAAI*. 2021.

[127] H.-S. Fang, J. Sun, R. Wang, M. Gou, Y.-L. Li, and C. Lu. "Instaboost: Boosting instance segmentation via probability map guided copy-pasting". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 682–691.

[128] C. Weitao and W. Zhibing. "Multi-level tail pixel cutmix and scale attention for long-tailed scene parsing". In: (July 2020). URL: https://openreview.net/forum?id=GHaQlkoNM-p.

[129] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao. "Deep High-Resolution Representation Learning for Visual Recognition". In: *CoRR* abs/1908.07919 (2019). arXiv: 1908.07919. URL: http://arxiv.org/abs/1908.07919.

[130] Q. Liu, B. B. Mohamadabadi, M. El-Khamy, and J. Lee. "Diversification is All You Need: Towards Data Efficient Image Understanding". In: (July 2020). URL: https://openreview.net/forum?id=UPbbSsBzfEW.

[131] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan. "AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty". In: *Proceedings of the International Conference on Learning Representations (ICLR)* (2020).

[132] Y. Yuan, X. Chen, and J. Wang. "Object-Contextual Representations for Semantic Segmentation". In: *CoRR* abs/1909.11065 (2019). arXiv: 1909.11065. URL: http://arxiv.org/abs/1909.11065.

[133] A. Shrivastava, A. Gupta, and R. Girshick. "Training Region-Based Object Detectors With Online Hard Example Mining". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[134] C.-C. Hsu and H.-T. Ma. "Edge-Preserving Guided Semantic Segmentation for VIPriors Challenge". In: *arXiv:2007.08919 [cs]* (July 2020). arXiv: 2007.08919. URL: http://arxiv.org/abs/2007.08919.

[135] S. Choi, J. T. Kim, and J. Choo. *Cars Can't Fly up in the Sky: Improving Urban-Scene Segmentation via Height-driven Attention Networks*. 2020. arXiv: 2003.05128 [cs.CV].

[136] V. B. Yesilkaynak, Y. H. Sahin, and G. Unal. "EfficientSeg: An Efficient Semantic Segmentation Network". In: *arXiv:2009.06469 [cs]* (Oct. 2020). arXiv: 2009.06469. URL: http://arxiv.org/abs/2009.06469.

[137] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, *et al.* "Searching for mobilenetv3". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 1314–1324.

[138] R. Pytel and T. Motyka. "Data-efficient semantic segmentation via extremely perturbed data augmentation". In: (July 2020). URL: https://vipriors.github.io/assets/downloads/Data-efficient%5C_semantic%5C_segmentation%5C_via%5C_extremely%5C_perturbed%5C_data%5C_augmentation.pdf.

[139] O. Ronneberger, P. Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi. Cham: Springer International Publishing, 2015, pp. 234–241. ISBN: 978-3-319-24574-4.

[140] *Progressive Sprinkles: A new data augmentation for CNN's*. https://lessw.medium.com/progressive-sprinkles-a-new-data-augmentation-for-cnns-and-helps-achieve-new-98-nih-malaria-6056965f671a. Accessed: 2024-07-02.

[141] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. "A Closer Look at Spatiotemporal Convolutions for Action Recognition". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6450–6459. DOI: 10.1109/CVPR.2018.00675.

[142] C. Feichtenhofer, H. Fan, J. Malik, and K. He. "SlowFast Networks for Video Recognition". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 6201–6210. DOI: 10.1109/ICCV.2019.00630.

[143] D. Tran, H. Wang, L. Torresani, and M. Feiszli. "Video Classification with Channel-Separated Convolutional Networks". In: *CoRR* abs/1904.02811 (2019). arXiv: 1904.02811. URL: http://arxiv.org/abs/1904.02811.

[144] C. Feichtenhofer. "X3D: Expanding Architectures for Efficient Video Recognition". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 200–210. DOI: 10.1109/CVPR42600.2020.00028.

[145] Z. Liu, X. Zhao, T. Huang, R. Hu, Y. Zhou, and X. Bai. "TANet: Robust 3D Object Detection from Point Clouds with Triple Attention". In: *AAAI* (2020). arXiv: 1912.05163 [cs.CV]. URL: https://arxiv.org/pdf/1912.05163.pdf.

[146] G. Bertasius, H. Wang, and L. Torresani. "Is Space-Time Attention All You Need for Video Understanding?" In: *arXiv preprint arXiv:2102.05095* (2021).

[147] H. Shao, S. Qian, and Y. Liu. "Temporal Interlacing Network". In: *AAAI* (2020).

[148] C. Yang, Y. Xu, J. Shi, B. Dai, and B. Zhou. "Temporal Pyramid Network for Action Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[149] Z. Liu, J. Ning, Y. Cao, Y. Wei, Z. Zhang, S. Lin, and H. Hu. "Video Swin Transformer". In: *arXiv preprint arXiv:2106.13230* (2021).

[150] T.-D. Truong, Q.-H. Bui, C. N. Duong, H.-S. Seo, S. L. Phung, X. Li, and K. Luu. "DirecFormer: A Directed Attention in Transformer Approach to Robust Action Recognition". In: *Computer Vision and Pattern Recognition*. 2022.

[151] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. "Temporal Segment Networks: Towards Good Practices for Deep Action Recognition". In: vol. 9912. Oct. 2016.

[152] Z. Tong, Y. Song, J. Wang, and L. Wang. "VideoMAE: Masked Autoencoders are Data-Efficient Learners for Self-Supervised Video Pre-Training". In: *Advances in Neural Information Processing Systems*. 2022.

[153] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. "Momentum Contrast for Unsupervised Visual Representation Learning". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 9726–9735. DOI: 10.1109/CVPR42600.2020.00975.

[154] C. Zach, T. Pock, and H. Bischof. "A Duality Based Approach for Realtime TV-L1 Optical Flow". In: *Pattern Recognition*. Ed. by F. A. Hamprecht, C. Schnörr, and B. Jähne. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 214–223. ISBN: 978-3-540-74936-3.

[155] K. Hara, H. Kataoka, and Y. Satoh. "Towards Good Practice for Action Recognition with Spatiotemporal 3D Convolutions". In: *2018 24th International Conference on Pattern Recognition (ICPR)*. 2018, pp. 2516–2521. DOI: 10.1109/ICPR.2018.8546325.

[156] J. Carreira and A. Zisserman. "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 4724–4733. DOI: 10.1109/CVPR.2017.502.

[157] H. Fan, B. Xiong, K. Mangalam, Y. Li, Z. Yan, J. Malik, and C. Feichtenhofer. "Multiscale Vision Transformers". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021.

[158] I. R. Dave, R. Gupta, M. N. Rizve, and M. Shah. "TCLR: Temporal Contrastive Learning for Video Representation". In: *arXiv preprint arXiv:2101.07974* (2021).

[159] J. Niu, Y. Gu, L. Nie, and C. You. "Semi-supervised Transformer with FPN for Bikes parts detection". In: (2021).

[160] K. C. Ishan Dave and M. Shah. ""Kallis" CRCV VIPriors challenge submission". In: (July 2020).

[161] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. "Learning Spatiotemporal Features with 3D Convolutional Networks". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 4489–4497. DOI: 10.1109/ICCV.2015.510.

[162] H. Chen, Z. Yu, X. Liu, W. Peng, Y. Lee, and G. Zhao. "2nd Place Scheme on Action Recognition Track of ECCV 2020 VIPriors Challenges: An Efficient Optical Flow Stream Guided Framework". In: (July 2020). URL: https://openreview.net/forum?id=R6YWiPVOQBo.

[163] B. Fernando, E. Gavves, J. Oramas M., A. Ghodrati, and T. Tuytelaars. "Rank Pooling for Action Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.4 (2017), pp. 773–787. DOI: 10.1109/TPAMI.2016.2558148.

[164] J. Lin, C. Gan, and S. Han. "TSM: Temporal Shift Module for Efficient Video Understanding". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 7082–7092. DOI: 10.1109/ICCV.2019.00718.

[165] T. Kim, H. Lee, M. Cho, H. Lee, D. h. Cho, and S. Lee. "Learning Temporally Invariant and Localizable Features via Data Augmentation for Video Recognition". In: (July 2020). URL: https://openreview.net/forum?id=tqz0rQvz_58.

[166] K. Q. Weinberger and L. K. Saul. "Distance metric learning for large margin nearest neighbor classification." In: *Journal of machine learning research* 10.2 (2009).

[167] Y. Sun, C. Cheng, Y. Zhang, C. Zhang, L. Zheng, Z. Wang, and Y. Wei. "Circle loss: A unified perspective of pair similarity optimization". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 6398–6407.

[168] Z. Zhong, L. Zheng, D. Cao, and S. Li. "Re-ranking person re-identification with k-reciprocal encoding". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1318–1327.

[169] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. "Total recall: Automatic query expansion with a generative feature model for object retrieval". In: *2007 IEEE 11th International Conference on Computer Vision*. IEEE. 2007, pp. 1–8.

[170] K. He, R. Girshick, and P. Dollár. "Rethinking imagenet pre-training". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4918–4927.

3

What Affects Learned Equivariance in Deep Image Recognition Models?

Equivariance w.r.t. geometric transformations in neural networks improves data efficiency, parameter efficiency and robustness to out-of-domain perspective shifts. When equivariance is not designed into a neural network, the network can still learn equivariant functions from the data. We quantify this learned equivariance, by proposing an improved measure for equivariance. We find evidence for a correlation between learned translation equivariance and validation accuracy on ImageNet. We therefore investigate what can increase the learned equivariance in neural networks, and find that data augmentation, reduced model capacity and inductive bias in the form of convolutions induce higher learned equivariance in neural networks.

This chapter has been published as:
R. Bruintjes, T. Motyka, J.C. van Gemert. "What Affects Learned Equivariance in Deep Image Recognition Models?". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2023, pp. 4839-4847.

3.1 INTRODUCTION

Equivariance in neural network features allows invariance to geometric transformations [1, 2], making such networks more data efficient [3–5], parameter efficient [2] and robust to out-of-distribution transformations [6–8].

Equivariance with respect to specific geometric transformations can be designed into the neural network architecture [2, 9, 10]. However, even with careful design, it may happen that the resulting architecture is not as equivariant as intended [7, 11–13]. An example is the convolution operator in Convolutional Neural Networks (CNNs) for translation equivariance, which can be broken by border effects [12, 13] or pooling [11]. On the other hand, even if neural networks are not designed to be equivariant, they can still *learn* equivariance naturally. Existing works demonstrate qualitative examples of learned equivariant features [14–16]. However, how much equivariance is learned, and which factors affect equivariance, are open questions.

In this work, we quantify learned equivariance in image recognition neural networks that have and have not been explicitly designed for equivariance. Where existing works [17–21] typically only measure equivariance at the output of the network, we measure equivariance for all intermediate layers. To do so, we deviate from existing measures of learned equivariance which are inconsistent across network depths, and we design a consistent measure.

Using our measure for learned equivariance, we find evidence that learned translation equivariance in intermediate features of neural networks correlates with increased validation accuracy on ImageNet. We therefore investigate how we can increase learned equivariance by changing how we train neural networks. In particular, we find that 1) making the task equivariant does not increased learned equivariance; 2) data augmentations designed for invariance indeed increase learned equivariance, even in early and middle layers; 3) reducing model capacity increases learned equivariance, suggesting that equivariant features arise from a need to compress representations; 4) CNNs learn more translation and rotation equivariance in intermediate features than the Vision Transformers (ViTs).

We make the following contributions:

- We propose a new measure for learned equivariance that allows comparing learned equivariance of features at different depths of the network.
- We show evidence for a positive correlation between learned translation equivariance in intermediate features and validation accuracy on ImageNet.

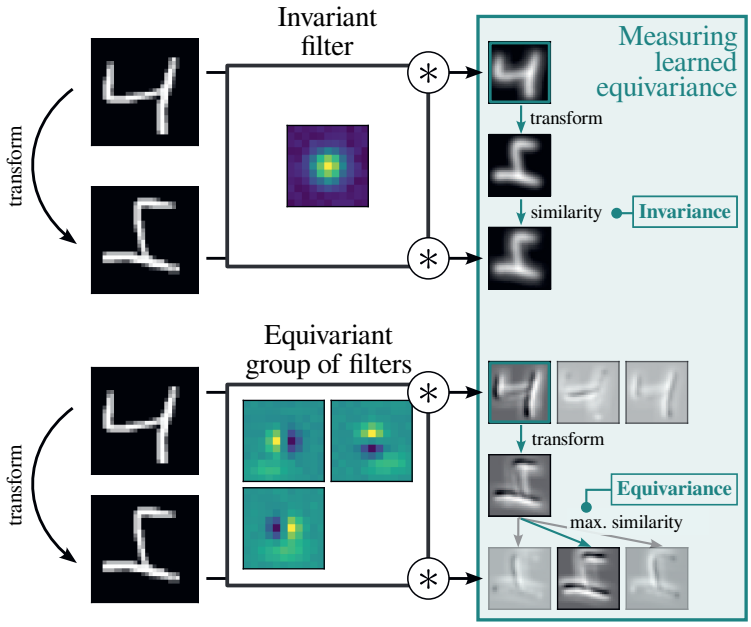


Figure 3.1: Neural networks can learn features that are invariant or equivariant w.r.t. a geometric transformation of the data, such as rotation. We measure learned equivariance w.r.t. translation and rotation in neural networks.

- We test how several aspects of neural network training affect learned equivariance. In summary, we find that data augmentation, reduced model capacity and the inductive biases of CNNs positively affect learned equivariance.

3.2 Related Works

Neural networks can learn equivariant features from data [16, 22, 23]. Particularly inspiring is the work by Olah *et al.* [16], that demonstrates by precise and meticulous manual investigations that learned equivariant features exist in networks that were not designed to be equivariant. Inspired by this work, we here investigate how to move beyond laborous manual qualitative investigations, and instead offer a quantitative approach, by giving an automatic measure for learned equivariance.

A number of existing works measure equivariance in neural networks. Goodfellow *et al*. [17] study models from the pre-Deep Learning era which have since been superseded by the models we study. More recent works measure equivariance in Convolutional Neural Networks, with KL divergence on class probabilities [18], with Euclidean distance [19], or cosine similarity [20, 21] on feature maps. In our work, we show that the cosine similarity is not appropriate for measuring equivariance in intermediate feature maps, and offer a correlation-based measure.

Several works study how neural network hyperparameters and datasets affect learned translation equivariance in the final output of the network. The kernel and padding sizes of the architecture affect translation invariance [21], although data augmentation might have a bigger effect on translation invariance than the network architecture [19]. Similar conclusions about the importance of the data were found by others [20, 24]. Here, we follow these investigations, and extend them by analyzing the impact on the intermediate layers.

There are some works that study equivariant properties of intermediate features. Recently, Moskalev *et al*. [25] proposed a method to detect invariance to any learned Lie group for intermediate features. However, they do not study equivariance, like we do. Other works study only the transformation group of translations ($\mathbb{Z}^2$). Zhang [11] measures the translation equivariance by computing cosine similarity between feature maps to show how max pooling violates the translation equivariance property. Kayhan *et al*. [12] and Islam *et al*. [13] show that some padding methods disrupt the translation equivariance property in CNNs. Ruderman *et al*. [26] measure the invariance of intermediate representations using normalized cosine similarity to study the effect of pooling on deformation stability. Where these works diagnose issues with designed equivariance and test for their effects, we consider learned equivariance in a more general sense, including transformation groups not designed into the network, such as rotations.

3.3 METHOD

Neural networks can learn to be equivariant in two ways: either by learning invariant features or by learning equivariant groups of features, as shown in Fig. 3.1. In this section we detail how we can measure the quantity of invariant features and equivariant groups of features. We discuss which similarity measure is appropriate for measurements of learned equivariance in features at different depths of a neural network. Finally, we verify our measures using artificially engineered equivariant CNNs.

In the following we will refer to invariant features and equivariant groups of features under the single predicate "learned equivariance", as invariance is a special case of equivariance.

3.3.1 Invariant features

We derive a measure of learned equivariance from inspecting the definition of equivariance [2] applied to a single neural network layer:

$$f(T_g(X)) = T'_g(f(X)), \tag{3.1}$$

where $X \in \mathbb{R}^{C_{\text{in}} \times H \times W}$ is an image or a feature map, $f(X) \in \mathbb{R}^{C \times H \times W}$ is the output of a neural network operation with C output features and T_g is the application of a transformation g from a transformation group G. For example, if $G = \mathbb{Z}^2$, then g is a translation with a particular integer-valued (x, y) offset. If T'_g is the identity function for all g, the layer f is *invariant* w.r.t. transformation T:

$$f(T(X)) = f(X). \tag{3.2}$$

Without designing invariance to T into neural network layer f, each individual feature in $f(X)$ can learn to behave invariant or not invariant with respect to T. We therefore define invariance for each feature $c \in C$ independently:

$$f(T(X))_c = f(X)_c. \tag{3.3}$$

In Fig. 3.1 we show an example where T is a 90° rotation.

To measure a feature's invariance w.r.t. g, we compute the similarity between $f(T(X))_c$ and $f(X)_c$:

$$\text{Invariance}(f_c, g) = S(f(T(X))_c, f(X)_c), \tag{3.4}$$

given a similarity function $S : \mathbb{R}^{\mathbb{H} \times \mathbb{W}} \times \mathbb{R}^{\mathbb{H} \times \mathbb{W}} \to [0, 1]$. Given invariance measures for each feature, we can average these measures for all features in a layer to compute a layer's invariance.

3.3.2 Equivariant features

A group of features $C_G \subseteq C$ in a neural network layer is equivariant with respect to a transformation group G if each feature $c' \in C_G$ activates for a different transformation g from the group. In other words, for a sample $T_g(X)$ transformed with any transformation from the group, the group of feature maps $f(X)_{c' \in C_G}$

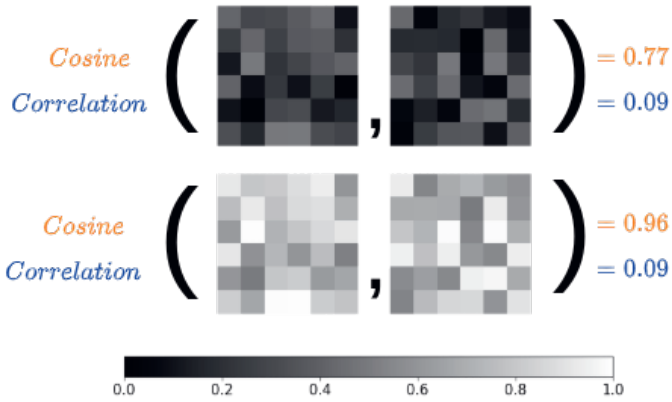


Figure 3.2: Comparison of different similarity measures on random feature maps. The values in feature maps on the bottom are 0.5 higher than the values in the top feature maps. Cosine similarity is sensitive to such shifts while correlation is invariant.

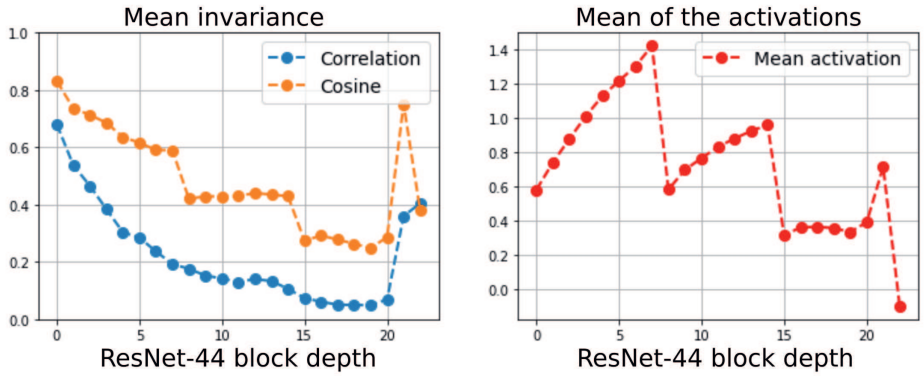


Figure 3.3: Comparison between magnitude of the activations (red) and equivariance computed using correlation (blue) and cosine similarity (orange). We compute correlation between the magnitude of the activations and equivariance scores computed with correlation (**0.11**) and cosine similarity (**0.63**).

will have one feature c' whose transformed feature map $T_g(f(X))_{c'}$ matches $f(T_g(X))_{c'}$:

$$f(T_g(X))_{c'} = T_g(f(X))_{c'}, \quad \exists c' \in C_T. \tag{3.5}$$

In Fig. 3.1 we show an example where g is a 90° rotation.

To fit this with the definition of equivariance (Eq. 3.1) we define T'_g:

$$f(T_g(X))_{c' \in C_G} = T'_g(f(X))_{c' \in C_G} \tag{3.6}$$

where T'_g transforms with T_g and selects feature c' that matches the transformation g. When this equation holds, the feature group C_G is equivariant w.r.t. T.

To measure the equivariance of a feature c we find the maximum similarity between $f(T_g(x))_c$ and $T_g(f(x))_{c'}$ over all features $c' \in C$, for a given transformation g:

$$\text{Equivariance}(f_c, g) = \max_{c' \in C} S(f(T_g(X))_c, T_g(f(X))_{c'}) \tag{3.7}$$

given a similarity function $S : \mathbb{R}^{\mathbb{H} \times \mathbb{W}} \times \mathbb{R}^{\mathbb{H} \times \mathbb{W}} \to [0, 1]$. Given equivariance measures for each feature, we can average these measures for all features in a layer to compute a layer's equivariance. Note that if a feature is invariant w.r.t. T_g, we will measure an equivariance score that is at least as high as the invariance score. As invariance is a special case of equivariance, this behavior of our measure is intended.

3.3.3 Measuring similarity

We need to choose a similarity measure $S : \mathbb{R}^{\mathbb{H} \times \mathbb{W}} \times \mathbb{R}^{\mathbb{H} \times \mathbb{W}} \to [0, 1]$ with which to compare feature maps to measure invariance and equivariance. In existing works, cosine similarity is commonly used as a similarity measure used to compute invariance or equivariance of the networks representations [11, 21, 24]. However, cosine similarity is sensitive to the mean values of its input vectors. This behaviour is depicted in Fig. 3.2. As different layers in a neural network have different mean activation values (see Fig. 3.3), this biases the similarity measure.

We propose to use Pearson correlation [27] instead. Pearson correlation, also called centered cosine similarity, is a similarity measure that does not suffer from sensitivity to the mean of the inputs, as it computes the covariance of the inputs

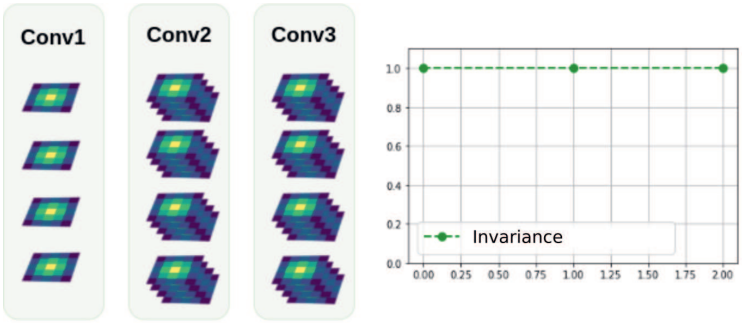


Figure 3.4: (left) Toy CNN for rotation invariance, designed to be perfectly invariant; (right) We measure perfect invariance using our method.

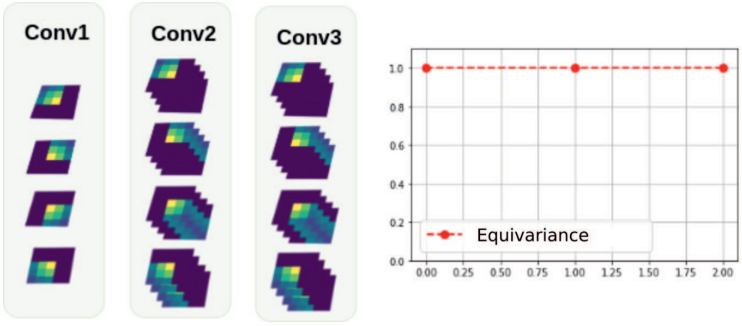


Figure 3.5: (left) Toy CNN for rotation equivariance, designed to be perfectly equivariant; (right) We measure perfect equivariance using our method.

normalized by their standard deviations. It is the basis of many methods for comparing network representations [28–30].

To motivate our choice, we visualize the difference between using cosine similarity and correlation for measuring equivariance in the following example. We train ResNet-44 [31] model on CIFAR-10 [32] dataset and compute invariance w.r.t. to 90° rotation after each residual block. In Figure 3.3 we show the qualitative comparison between the scores, computed using cosine similarity and correlation, and the mean of the activations. Additionally, we compute a correlation between the magnitude of the activations and equivariance scores computed with cosine similarity (**0.63**) and correlation (**0.11**). Scores computed using cosine similarity correlate visibly with the mean of the activation while,

for the scores computed using correlation, this effect is less prevalent. In our experiments, we therefore use correlation as a measure to quantify equivariance.

3.4 Experiments

3.4.1 Controlled experiments

To verify that our method captures equivariance, we apply it to two controlled toy settings. We create two 3-layer CNNs with hand-crafted filters such that we expect to measure perfect learned rotation invariance and equivariance respectively. For the invariant model, we set all the filters to be rotationally symmetric, using a 2D isotropic Gaussian function, and measure the invariance after each layer (Fig. 3.4). For the equivariant model, we cut out corners of the filters from the invariant model such that all the filters are rotations of one another (Fig. 3.5). Our measure finds both models capture exactly the intended learned equivariances, demonstrating the validity of our measure.

3.4.2 Does learned equivariance improve accuracy?

We study the relationship between the validation accuracy and the amount of learned equivariance in large-scale seminal models. For each part of each trained model we compute Spearman's rank correlation between the amount of invariance or equivariance and the ImageNet validation accuracy of the model.

We test four CNNs (EfficientNet-B6 & EfficientNet-B7 [33], ResNeXT-101 [34] and Inception-V3 [35]) and two Vision Transformer variants (Vision Transformer [36] and MLP-Mixer [37]). We measure invariance and equivariance for both translation and rotation for 2000 images from the ImageNet validation set. We do not train the models ourselves but instead use available checkpoints from `torchvison` [38] or `timm` [39]. Since the studied model families do not have the same exact number of layers, we divide each model into depth-wise parts and report the average equivariance measures over all layers in each part. Feature maps from the beginning of the network until the global average pooling (GAP) layer are uniformly partitioned into *Early*, *Middle* and *Late* parts. *Pool* captures the feature maps directly after the GAP layer and *Final* is the feature map directly before the softmax layer. We discriminate between the *Pool* part and the *Final* part to identify what role in achieving equivariance the global pooling and final classifier have.

Fig. 3.6 shows there is some correlation between translation equivariance in

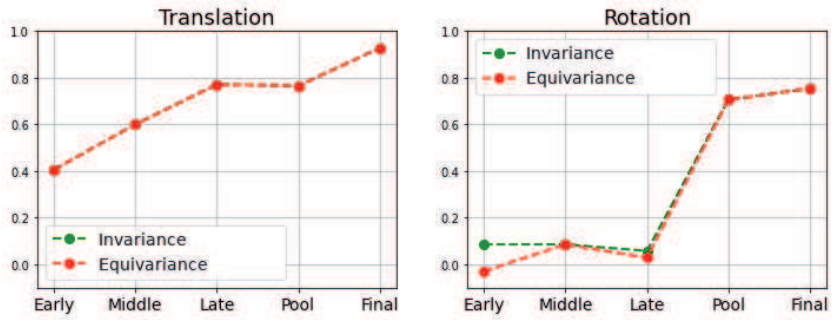


Figure 3.6: Spearman's rank correlation between learned equivariance and ImageNet validation accuracy. Translation equivariance in intermediate features correlates with increased accuracy on ImageNet, while rotation equivariance does not.

Early and *Middle* layers and accuracy on ImageNet, while attaining almost perfect correlation in the *Final* part. In contrast, for rotations there is little correlation between the equivariance in the representation before global pooling and the validation accuracy.

Even though the sample size (six models) for this correlation test is small, we conclude that there is some evidence for the benefit of learning translation equivariance in intermediate features of neural networks trained on ImageNet. In the following we therefore study what can increase the learned equivariance in such networks.

3.4.3 Equivariance in the Data

On tasks where invariant responses are beneficial to solve the task, e.g. translation invariance in image recognition, one may wonder how this invariance is achieved. We study how learned equivariance in intermediate features is affected by adding transformations to the data and therefore into the task. We choose to study rotation transformations on CNNs, as rotation equivariance is not designed into CNNs. We study whether there is a difference if the task is invariant or equivariant with respect to introduced transformations.

We train a 7-layer CNN taken from Cohen *et al*. [2], consisting of 7 layers of 3×3 convolutions, 20 channels in each layer, ReLU activation functions, batch normalization, and max-pooling after layer 2, on 3 different datasets. We test on three different datasets. The first dataset is *MNIST6*, which is the regular MNIST [40] without $\{0, 1, 6, 8\}$ classes, to get rid of rotational transformations

that these classes have. For example, digit 8 is very similar to its 180° rotation, so, by default, this class would introduce some rotation invariance, which is undesirable as we want to control for rotation invariance in this setting. Second is the *MNIST6-Rot-Inv* where every digit in *MNIST6* is randomly rotated by $r \in \{0°, 90°, 180°, 270°\}$ upfront. This dataset imposes invariance into the task as, for every transformation, the predicted class should be the same. The last dataset, *MNIST6-Rot-Eq*, is created in the same way as *MNIST6-Rot-Inv*, but now the classes are made up of all combinations of digit number and rotation (e.g. class 2 is $(\text{digit}\,0, 180°)$). This dataset imposes equivariance into the task. We compute in- and equivariance of the trained model for 2000 images from the validation set. We average the score over $90°, 180°, 270°$ rotations. Each experiment is repeated three times using different random seed. We train for 100 epochs using Adam with a batch size of 128 and a learning rate of 0.01, L2 regularization at 0.0005 and weight decay at epochs 25 and 50 with a factor of 10.0.

In Fig. 3.7 we show the learned rotation equivariance. Firstly, we observe that the equivariance decreases with the depth, up to the final part after global average pooling (GAP), regardless of the task. For the tasks where the equivariance or invariance is imposed in the task, we see an increase in the final part, which suggests that GAP plays a significant role in achieving equivariance. Secondly, we do not see any significant differences between *MNIST6*, *MNIST6-Rot-Inv* and *MNIST6-Rot-Eq*, up to a later stage, which may indicate that early convolutional layers learn features with some amount of rotation equivariance regardless of the rotation invariance of the task. Finally, we observe that rotation equivariance is much larger then rotation invariance in the early and middle layers, which shows that CNNs do learn more rotated versions of the same feature in different channel rather then learning invariant, symmetrical features in a single channel. We conclude that introducing equivariance into the task does not significantly affect the learned equivariance of intermediate features.

3.4.4 Data augmentations

By duplicating input samples under some transformation, data augmentation can induce invariance in the neural network. We study what the effect of data augmentation is on learned equivariance in intermediate representations: does data augmentation result in more invariant or equivariant intermediate features? For example, we know that the random crops data augmentation method, which essentially introduces random translations into the data, increases the model performance and translation invariance at the last layer [19]. The question is whether random crops increase the learned equivariance of intermediate features

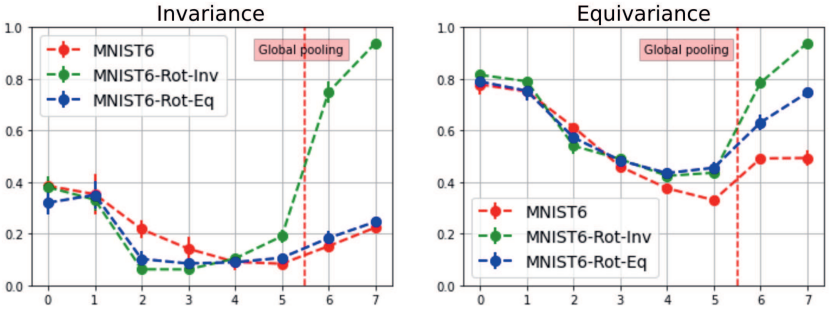


Figure 3.7: Learned rotation equivariance for rotation invariance/equivariance in the data. Invariance or equivariance in the task does not induce learning more equivariant features up until the late part of the network. Also there is no visible difference, up until the late part of the network, in the learned equivariance between the invariant and equivariant tasks.

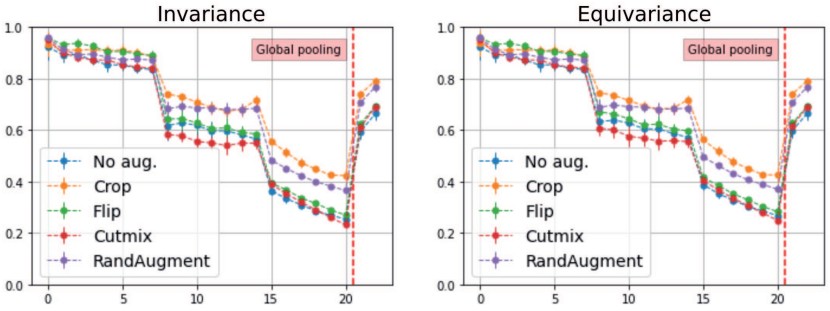


Figure 3.8: Measuring learned translation equivariance for data augmentations. Random crops and RandAugment increase channel equivariance the most, while other strategies have no discernible improvements.

as well. In this experiment we study how translation equivariance is affected by different data augmentations.

We train ResNet-44 [31], adapted for CIFAR-10 [32], on the CIFAR-10 dataset using one of the following augmentations: random crops, horizontal flips, CutMix [41], RandAugment [42]. In each experiment we compute equivariance of the trained model over 2000 images from the validation set and average the score over diagonal shifts from one to 16 pixels. Each experiment is repeated three times by training the network different random seeds. We train for 200 epochs

using SGD with a batch size of 128 and a learning rate of 0.1 and a momentum of 0.9, L2 regularization at 0.0001 and weight decay at epochs 100 and 150 with a factor of 10.0.

In principle we expect the in- and equivariance to be the same since translation equivariance should be provided by the convolution. However, we include equivariance in our experiments since there are works showing that the information about location can be encoded in different channels [12, 13].

In Fig. 3.8 we show learned translation equivariance for the tested data augmentations. Random crops and RandAugment increase the equivariance of learned features in the *Middle*, *Late* and *Final* parts, while the other data augmentation methods do not have any significant effect, with CutMix even having less equivariance than the baseline in the *Middle* part. We complement the finding of Kauderer-Abrams [19] by showing that random crops increase not only translation invariance but also translation equivariance in the intermediate layers. Also, we do not see any difference between invariance and equivariance for any data augmentation, which means that any equivariance learned is just invariance.

3.4.5 Model capacity

We hypothesize that a smaller model in principle benefits from a more efficient representation and hence may learn more equivariant features. We therefore study whether model capacity influences learning translation equivariant representations. We train WideResNet-40 (WRN-40) [43] models, where we scale the number of channels (the "width") in each layer by a factor $s \in \{1, 2, 4, 8\}$. We train on the CIFAR-10 dataset and measure learned translation equivariance. The hyperparameters used for training are the same as in the data augmentation experiment of Section 3.4.4.

In Fig. 3.9 we show learned translation equivariance for different model capacities. We observe that the amount of translation equivariance is lower for the wider models, even though the amount of invariance in the final part is the same, which matches our hypothesis: an efficient representation learns to be equivariant.

3.4.6 Architectures

The architecture of a neural network determines which biases can be learned in training. Vision Transformers (ViTs) [36] lack certain inductive biases present in CNNs, which has been linked to their reduced data efficiency [44, 45]. We are

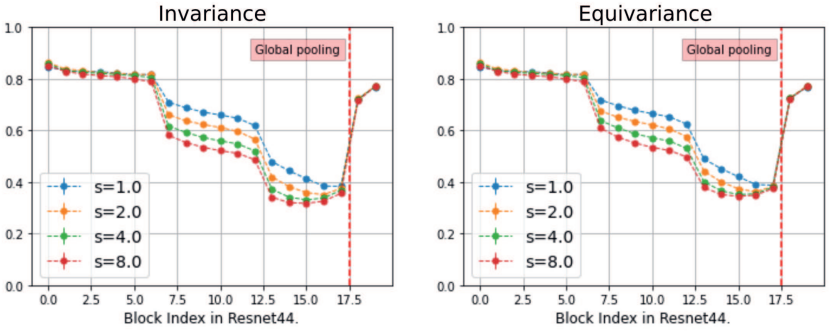


Figure 3.9: Measuring learned translation equivariance for model capacity with channels *s*. Smaller models learn more in- and equivariance, although the amount of in- and equivariance in the end is similar.

interested to what extent the difference in inductive bias between CNNs and Vision Transformers (ViTs) affects learned equivariance.

We test architectures as they were designed for the ImageNet dataset [46], to faithfully represent their intended inductive bias. We use the same architectures and pre-trained model weights as tested in Section 3.4.2: four CNNs (EfficientNet-B6 & EfficientNet-B7 [33], ResNeXT-101 [34] and Inception-V3 [35]) and two Vision Transformer variants (Vision Transformer [36] and MLP-Mixer [37]). We measure both translation and rotation equivariance on trained models for 2000 images from the ImageNet validation set. We also use the same depth-wise partitioning of feature maps into parts as used in Section 3.4.2. We measure translation equivariance over diagonal shifts of size 1 to 32 and rotation equivariance for 90°, 180°, 270° rotations.

In Figure 3.10 we present the results for learned translation equivariance. We can see that ViT and MLP-Mixer have less translation equivariance than CNNs in *Early* and *Middle* layers. This is not unexpected, as convolutions directly integrate translation equivariance, whereas Vision Transformers have to learn position embeddings that are translation equivariant. This reduced translation equivariance could be the reason for the poor data efficiency of ViT and MLP-Mixer [36, 37] since translation equivariance improves data efficiency [12]. Finally, we note that learned invariance and equivariance are identical for the tested models, meaning that these networks do not learn to represent different translations in different channels.

In Fig. 3.11 we present the results for learned rotation equivariance. We observe that the ViT and MLP-Mixer have lower rotation equivariance than the CNNs

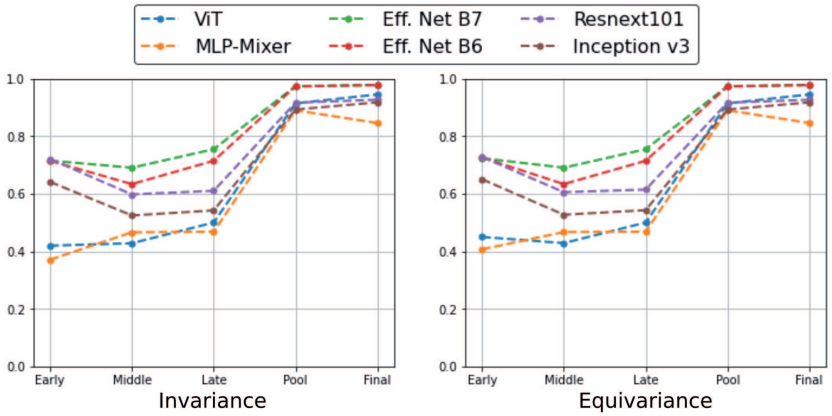


Figure 3.10: Measuring learned translation equivariance for inductive biases. The CNN variants exhibit more equivariance in the intermediate representation then the Vision Transformer variants. Global pooling seems to play an important role in achieving invariance.

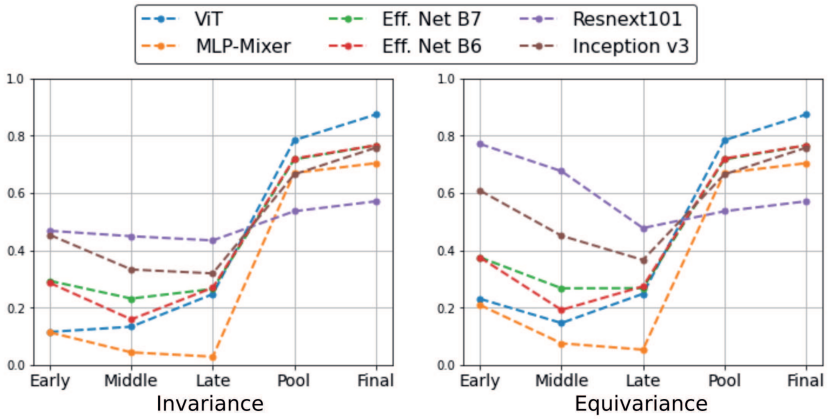


Figure 3.11: Measuring learned rotation equivariance for inductive biases. The CNN variants exhibit more equivariance in the intermediate representation then the Vision Transformer variants. The *Early* and *Middle* parts have more equivariance than invariance.

in intermediate features, while after the GAP layer the ViT exhibits the most rotation equivariance out of all the models. Secondly, we note that early parts of all networks learn equivariant features that are not invariant, more so than in late parts of the networks. In contrast to the results for translation equivariance, we see that models with low rotation equivariance throughout *Early*, *Middle* and *Late* parts (ViT, Efficient-Net B6/B7) have the highest rotation equivariance in the *Final* part, while the models with highest equivariance in *Early*, *Middle* and *Late* parts (ResNeXT-101, Inception-v3) have the least equivariance in the *Final* part. This shows that high learned equivariance in the final model representation does not imply that intermediate representations are also highly equivariant.

3.5 Conclusion

We conduct a quantitative study on learned equivariance in intermediate features of CNNs and Vision Transformers trained for image recognition, using an improved measure of equivariance. We find evidence that translation equivariance in intermediate representations correlates with ImageNet validation accuracy. We show that data augmentations and reduced model capacity can increase learned equivariance in intermediate features. Also, the CNNs we test learn more translation and rotation equivariance in intermediate features than the ViTs we test.

Limitations. Our method allows to measure equivariance w.r.t. affine transformations only. The reason for that is the transformation g with respect to which we measure the equivariance has to be a map from and to an identical discrete domain, e.g. feature maps. This restriction disqualifies continuous transformations such as rotations with any other resolution than 90 degrees, or scaling with non-integer scaling factors.

Future work. Learned equivariance benefits image recognition models. However, applying equivariant priors usually adds additional cost in terms of memory or computation. Future work could study whether one can apply equivariant priors selectively within a neural network, saving computing cost where networks already learn to be equivariant. Additionally, we show that Vision Transformers learn less translation equivariance than CNNs. Future work could explore methods to increase translation invariance in Vision Transformers, to aid in their data efficiency.

REFERENCES

[1] Y. LeCun, Y. Bengio, *et al.* "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995.

[2] T. S. Cohen and M. Welling. "Group Equivariant Convolutional Networks". In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16. New York, NY, USA: JMLR.org, 2016, pp. 2990–2999.

[3] M. Weiler, F. A. Hamprecht, and M. Storath. "Learning steerable filters for rotation equivariant cnns". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 849–858.

[4] M. Winkels and T. S. Cohen. "3D G-CNNs for pulmonary nodule detection". In: *arXiv preprint arXiv:1804.04656* (2018).

[5] A. Lengyel and J. van Gemert. "Exploiting learned symmetries in group equivariant convolutions". In: *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2021, pp. 759–763.

[6] T. Altstidl, A. Nguyen, L. Schwinn, F. Köferl, C. Mutschler, B. Eskofier, and D. Zanca. "Just a Matter of Scale? Reevaluating Scale Equivariance in Convolutional Neural Networks". In: *arXiv preprint arXiv:2211.10288* (2022).

[7] T. Edinxhoven. "Using and Abusing Equivariance". MA thesis. Delft University of Technology, 2023.

[8] M. Sangalli, S. Blusseau, S. Velasco-Forero, and J. Angulo. "Scale Equivariant U-Net". In: *arXiv preprint arXiv:2210.04508* (2022).

[9] M. Weiler and G. Cesa. "General e (2)-equivariant steerable cnns". In: *Advances in Neural Information Processing Systems* 32 (2019).

[10] T. Cohen, M. Weiler, B. Kicanaoglu, and M. Welling. "Gauge equivariant convolutional networks and the icosahedral CNN". In: *International conference on Machine learning*. PMLR. 2019, pp. 1321–1330.

[11] R. Zhang. "Making Convolutional Networks Shift-Invariant Again". In: *CoRR* abs/1904.11486 (2019). arXiv: 1904.11486. URL: http://arxiv.org/abs/1904.11486.

[12] O. S. Kayhan and J. C. v. Gemert. "On translation invariance in cnns: Convolutional layers can exploit absolute spatial location". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 14274–14285.

[13] M. A. Islam, M. Kowal, S. Jia, K. G. Derpanis, and N. D. Bruce. "Global pooling, more than meets the eye: Position information is encoded channel-wise in cnns". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 793–801.

[14] J. Bergstra, A. Courville, and Y. Bengio. "The statistical inefficiency of sparse coding for images (or, one Gabor to rule them all)". In: *arXiv preprint arXiv:1109.6638* (2011).

[15] S. Dieleman, J. De Fauw, and K. Kavukcuoglu. "Exploiting cyclic symmetry in convolutional neural networks". In: *International conference on machine learning*. PMLR. 2016, pp. 1889–1898.

[16] C. Olah, N. Cammarata, C. Voss, L. Schubert, and G. Goh. "Naturally occurring equivariance in neural networks". In: *Distill* 5.12 (2020), e00024–004.

[17] I. Goodfellow, H. Lee, Q. Le, A. Saxe, and A. Ng. "Measuring invariances in deep networks". In: *Advances in neural information processing systems* 22 (2009).

[18] A. Zhou, F. Tajwar, A. Robey, T. Knowles, G. J. Pappas, H. Hassani, and C. Finn. "Do Deep Networks Transfer Invariances Across Classes?" In: *arXiv preprint arXiv:2203.09739* (2022).

[19] E. Kauderer-Abrams. *Quantifying Translation-Invariance in Convolutional Neural Networks*. 2017. arXiv: 1801.01450 [cs.CV].

[20] V. Biscione and J. Bowers. "Learning translation invariance in CNNs". In: *arXiv preprint arXiv:2011.11757* (2020).

[21] J. C. Myburgh, C. Mouton, and M. H. Davel. "Tracking translation invariance in CNNs". In: *Artificial Intelligence Research: First Southern African Conference for AI Research, SACAIR 2020, Muldersdrift, South Africa, February 22-26, 2021, Proceedings 1*. Springer. 2020, pp. 282–295.

[22] K. Lenc and A. Vedaldi. "Understanding image representations by measuring their equivariance and equivalence". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 991–999.

[23] C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, and S. Carter. "An overview of early vision in inceptionv1". In: *Distill* 5.4 (2020), e00024–002.

[24] D. Bouchacourt, M. Ibrahim, and A. S. Morcos. "Grounding inductive biases in natural images: invariance stems from variations in data". In: *arXiv preprint arXiv:2211.10288* (2021). arXiv: 2106.05121 [cs.CV].

[25] A. Moskalev, A. Sepliarskaia, I. Sosnovik, and A. Smeulders. "LieGG: Studying Learned Lie Group Generators". In: *arXiv preprint arXiv:2210.04345* (2022).

[26] A. Ruderman, N. C. Rabinowitz, A. S. Morcos, and D. Zoran. "Pooling is neither necessary nor sufficient for appropriate deformation stability in CNNs". In: *arXiv preprint arXiv:1804.04438* (2018).

[27] D. Freedman, R. Pisani, and R. Purves. "Statistics (international student edition)". In: *Pisani, R. Purves, 4th edn. WW Norton & Company, New York* (2007).

[28] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton. "Similarity of neural network representations revisited". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 3519–3529.

[29] M. Kuss and T. Graepel. "The geometry of kernel canonical correlation analysis". In: (2003).

[30] M. Raghu, J. Gilmer, J. Yosinski, and J. Sohl-Dickstein. "Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability". In: *Advances in neural information processing systems* 30 (2017).

[31] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[32] A. Krizhevsky, G. Hinton, *et al.* "Learning multiple layers of features from tiny images". In: (2009).

[33] M. Tan and Q. Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International conference on machine learning*. PMLR. 2019, pp. 6105–6114.

[34] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. "Aggregated residual transformations for deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1492–1500.

[35] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[36] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.* "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[37] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, *et al.* "Mlp-mixer: An all-mlp architecture for vision". In: *Advances in neural information processing systems* 34 (2021), pp. 24261–24272.

[38] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[39] R. Wightman. *PyTorch Image Models*. https://github.com/rwightman/pytorch-image-models. 2019. DOI: 10.5281/zenodo.4414861.

[40] L. Deng. "The mnist database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[41] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. "Cutmix: Regularization strategy to train strong classifiers with localizable features". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 6023–6032.

[42] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le. "Randaugment: Practical Automated Data Augmentation With a Reduced Search Space". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2020.

[43] S. Zagoruyko and N. Komodakis. "Wide residual networks". In: *arXiv preprint arXiv:1605.07146* (2016).

[44] Y. Xu, Q. Zhang, J. Zhang, and D. Tao. "Vitae: Vision transformer advanced by exploring intrinsic inductive bias". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 28522–28535.

[45] S. d'Ascoli, H. Touvron, M. L. Leavitt, A. S. Morcos, G. Biroli, and L. Sagun. "Convit: Improving vision transformers with soft convolutional inductive biases". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 2286–2296.

[46] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

4

Learning to Adapt to Position Bias in Vision Transformer Classifiers

How discriminative position information is for image classification depends on the data. On the one hand, the camera position is arbitrary and objects can appear anywhere in the image, arguing for translation invariance. At the same time, position information is key for exploiting capture/center bias, and scene layout, e.g.*: the sky is up. We show that position bias, the level to which a dataset is more easily solved when positional information on input features is used, plays a crucial role in the performance of Vision Transformers image classifiers. To investigate, we propose Position-SHAP, a direct measure of position bias by extending SHAP to work with position embeddings. We show various levels of position bias in different datasets, and find that the optimal choice of position embedding depends on the position bias apparent in the dataset. We therefore propose Auto-PE, a single-parameter position embedding extension, which allows the position embedding to modulate its norm, enabling the unlearning of position information. Auto-PE combines with existing PEs to match or improve accuracy on classification datasets.*

This chapter has been published as:
R. Bruintjes and J. C. van Gemert. "Learning to Adapt to Position Bias in Vision Transformer Classifiers". In: *arXiv:2505.13137 [cs.CV], 2025.*

Code available at:
https://github.com/rjbruin/position-shap

4.1 INTRODUCTION

Vision Transformers [1] explicitly infuse their learned representations with information on the position of features using position embeddings. This goes against the long-held belief that, when trained specifically for image classification, the learned representations of vision models like Convolutional Neural Networks should be free of positional information (e.g. be translation invariant) [2–5]. However, adding position embeddings often improves the performance of Vision Transformers significantly in image classification [1, 6–10]. This raises the questions *how* and *why* position information affects Vision Transformers. This work investigates these questions.

We define *position bias* in datasets as the level to which knowing the spatial position of objects of interest in the input images of the dataset helps the model to learn to classify the data. An example of a particularly common source of position bias is *capture bias* [11, 12], which occurs when the object of interest is commonly framed in the center of the photograph. When multiple objects are present, the model should learn to only classify the center object, and therefore needs to use information on the position of input features (see Fig. 4.1). Other position biases are even more overt, such as the bias apparent in the SVHN [13] dataset, where the classification task is explicitly to only classify the center digit in the image. There are also datasets that theoretically do not contain any position bias, such as EuroSAT [14, 15], which is comprised of satellite imagery where the camera position is independent of the content. We conclude that the level of position bias in classification datasets varies, and hypothesize that the ability of classification models to model this position bias affects the accuracy of these models.

In this work, we explore the effects of varying position bias in classification datasets on the accuracy of Vision Transformers. First, to know the level of position bias in a dataset, we should measure it. However, no direct measurement method exists to measure position bias. Therefore, we propose **Position-SHAP** (P-SHAP), a direct measure of position bias learned by classification models.

Using P-SHAP, we measure position bias in models trained on several classification datasets and find that models trained on datasets with high position bias do better when learning more position bias. Furthermore, we show that using position-affecting augmentations (such as RandAugment [16]) and transfer learning affect the learned position bias.

Finally, we show that knowing the position bias of a dataset informs the optimal choice of position embedding method. Specifically, we find that models trained on datasets with low position bias perform better without a position embedding than with a learnable absolute position embedding (APE). We there-

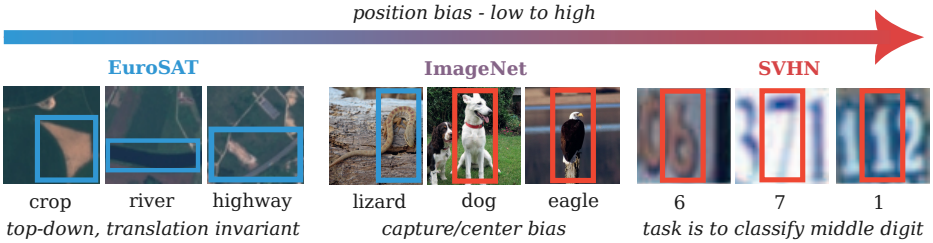


Figure 4.1: In addition to appearance information, some classification datasets require position information to be solved, and some do not. In EuroSAT, the object of interest appears anywhere in the picture. In ImageNet, often but not always the object of interest is centered in the frame ("capture bias"). In SVHN, the task is explicitly to classify only the middle digit.

fore introduce a simple single-parameter extension to position embeddings that learns to reduce the norm of the position embedding, called **Auto-PE**. Adding Auto-PE to RoPE [17] position embeddings matches or improves performance in all tested settings, without having to manually tune the method of position embedding.

All code and experiments are made publicly available at https://github.com/rjbruin/position-shap.

4.2 Related work

Vision Transformers and position embeddings. Vision Transformers [1] apply transformers [18] to computer vision tasks. To make input images compatible with transformers, images are divided into flattened patches, whose dimensionalities are reduced by a linear mapping to match the hidden layers of the transformer, before position embeddings (PEs) are added:

$$\mathbf{X}^P = \text{Patchify}(\mathbf{X}), \qquad \mathbf{X} \in \mathbb{R}^{H \times W \times C}, \mathbf{X}^P \in \mathbb{R}^{N_P \times (H_P \times W_P \times C)} \tag{4.1}$$

$$\mathbf{Y}_i = \text{Linear}(\mathbf{X}_i^P) + \mathbf{P}_i, \qquad i \in N_P, \mathbf{P} \in \mathbb{R}^{N_P \times D}, \mathbf{Y} \in \mathbb{R}^{N_P \times D} \tag{4.2}$$

where $\mathbf{X}$ is a single input image, $H \times W \times C$ are the image dimensions, N_P is the number of patches, $H_P \times W_P$ are the spatial dimensions of the patches, $\mathbf{X}^P$ are the patches of the input sample, $\mathbf{P}$ is the position embedding, and $\mathbf{Y}$ is the input to the first transformer block. Since transformers are permutation-invariant

with respect to the patch dimension, learned representations cannot know their position without PEs.

There are many different PE methods. We distinguish baseline methods, introduced during the initial development of ViTs, from original contribution methods, published in later years.

Baseline PEs. The Absolute Position Embedding (APE) [18] learns a one-dimensional token for each input patch, which is initialized randomly. Several alternative methods initialize the APE using a fixed function (e.g. sinusoid-based [18]), and continue to either fine-tune the APE or keep it fixed during training. Alternatively, Relative Position Embedding (RPE) [19] methods infuse position information into the self-attention operation, by adding biases to the attention scores dependent on the relative position of tokens.

Original contribution PEs. Learnable Fourier Features [20] generalize the APE method initialized with sinusoidal features to multiple dimensions and learn the frequencies of the wave functions end-to-end. CAPE [21] augments APE embeddings to reduce absolute positional information of features, while retaining relative positional information between features. iRPE [22] extends the RPE method to allow for the bias term to be dependent on the input tokens, and groups relative distances in the computation of this bias according to their size. Rotary Position Embedding (RoPE) [17, 23] integrates into the attention operation, like RPE, where it transforms incoming tokens to complex values by reshaping, before rotating the complex-valued input tokens around the origin by end-to-end learned angles, before transforming the tokens back to real values. Because the works proposing these models show improved accuracy, we analyze these PE methods for their effect on the position bias directly using P-SHAP.

Position bias. In the era before Vision Transformers, Convolutional Neural Networks (CNNs) dominated computer vision tasks. CNNs are designed to be translation invariant [24] so as to share gradients among translated input features. As a result, they should not be able to model the position of input features. However, border effects [2, 3] and improper subsampling [4, 5] allow CNNs to learn positional information. Several works aim to remove position information by improving translation invariance in CNNs [3, 4, 25] and in Vision Transformers [26], with improved performance as a result. However, adding position embeddings to Vision Transformers, thereby enabling models to learn positional information, improves performance as well. This work unifies these findings by showing that position information benefits performance on

classification datasets that contain position bias, but does not benefit performance on datasets without position bias.

Related notions. Various works have explored notions similar to position bias, coining terms such as "positional variation" [27], "structural learning" [28], "position difference" [7], "local processing" [7], or "token mixing" [29]. Some of these works also perform measurements on these notions, such as on locality (the distance between patch representations in the attention operation) [7, 9]. However, any measure of something other than position bias may be affected by confounding factors. For example, using shifted pixels to measure change in accuracy, as in Zhang [4], can show the effect of position bias but is confounded by the effects of the patch size of the model, as shifting the image by a number of pixels equal to the patch size will yield identical patches. We therefore propose a direct measure of position bias, unaffected by confounding factors.

Testing performance. Some works perform perturbation analysis and measure effects on accuracy [4, 30]. Other works show improved performance by embedding inductive biases [8, 31–33] into models or using pretext tasks [34–36], aiming to improve position bias. These results suggest position bias is important and should be investigated. However, these works do not measure position bias and can therefore not conclusively demonstrate that their improved results are due to position bias.

Kernel SHAP and Vision Transformers. Our measure of position bias builds on Kernel SHAP [37]. Kernel SHAP is a feature attribution method, which means it assigns importance scores ("SHAP values") to each input feature in a single input sample $\mathbf{X}$ of a neural network. Kernel SHAP assumes that the network response $f(\mathbf{X})$ can be approximated by a linear function $g(\mathbf{X})$:

$$f(\mathbf{X}) \approx g(\mathbf{X}) = \phi_0^{\mathbf{X}} + \sum_{i}^{H} \sum_{j}^{W} \sum_{k}^{C} \phi_{ijk}^{\mathbf{X}} \mathbf{X}_{ijk} \tag{4.3}$$

where $\boldsymbol{\phi}^{\mathbf{X}} \in \mathbb{R}^{HWC+1}$ are the SHAP values that represent the impact of each input feature in $\mathbf{X}$, with $\phi_0^{\mathbf{X}}$ representing the impact of all collected biases of the network.

Estimating SHAP values. The SHAP values $\boldsymbol{\phi}^{\mathbf{X}}$ are estimated for each sample $\mathbf{X}$ by weighted linear regression. The loss function depends on the difference between the actual network response $f(\mathbf{X})$, and f applied to versions of $\mathbf{X}$ where the values of all possible subsets of features are replaced with a *background value*, to emulate removal of features. This background value is the mean value of the feature in a so-called *background dataset*, which should have the same distribution

as the sample, *e.g.*the training dataset. See Section 4.A for details.

Kernel SHAP for PEs. In this work we would like to use Kernel SHAP to estimate SHAP values for PEs. However, the values of the PE are the same for all samples in any background dataset, as they are a bias parameter and therefore constant w.r.t. the input (see Eq. 4.2 where $\mathbf{P}$ does not depend on $\mathbf{X}$). If the background value is equal to the value of the feature, the Kernel SHAP loss is zero valued, and the SHAP value cannot be estimated. Therefore, it is impossible to estimate SHAP values for the PE using Kernel SHAP as is. However, in this work, we propose Position-SHAP, an extension of Kernel SHAP that *can* attribute SHAP values to position embeddings.

4.3 Position-SHAP

For Kernel SHAP to compute SHAP values for the PE, we need to choose a suitable way to set the background value of the PE. To recap, for input features, the appropriate background dataset has the same distribution as the input feature. Our contribution is to consider each PE token $\mathbf{P}_i \in \mathbb{R}^D$ in the PE to be a sample from a distribution of PE tokens learned by this model, and to consider the set of all PE tokens $\mathbf{P}$ as the background dataset. In practice, we extend the implementation of Kernel SHAP to include the PE as a feature by shuffling the PE along the token dimension when computing background values.

We are now able to compute SHAP values for each element of the PE. However, we would like to have a single SHAP value to represent the impact of the PE. Fortunately, because g is a linear function, the SHAP value of a group of features is the sum of the SHAP values of the individual features. We can therefore compute a single SHAP value for the group of features.

Finally, **Position-SHAP** (P-SHAP) is defined as the ratio between the SHAP value of the PE and the SHAP value of the PE and image together:

$$g(\mathbf{X}) = \phi_0^{\mathbf{X}} + \phi_P^{\mathbf{X}} \sum \mathbf{P} + \phi_I^{\mathbf{X}} \sum \mathbf{X} \tag{4.4}$$

$$\text{Position-SHAP}(\mathbf{X}) = \frac{\phi_P^{\mathbf{X}}}{\phi_P^{\mathbf{X}} + \phi_I^{\mathbf{X}}}, \tag{4.5}$$

where $\phi_P^{\mathbf{X}}$ is the SHAP value for the PE and $\phi_I^{\mathbf{X}}$ is the SHAP value for the input image.

We compute P-SHAP for individual samples. In our experiments, the P-SHAP value of a model over a test set is taken as the average over the correctly classified

samples in the test set. We note that Kernel SHAP (and therefore P-SHAP) has quite a high compute cost. We discuss additional hyperparameters as well as inference cost and consistency of P-SHAP in Section 4.E.

4.4 AUTO-PE

Inspired by our findings on the importance of position bias, we propose **Auto-PE**, a simple single-parameter method that learns to reduce or increase the impact of the position embedding by modulating the norm of the position embedding using a learned weight γ, before adding the position embedding to the patch embeddings:

$$\mathbf{Y}_i = \text{Linear}(\mathbf{X}_i^p) + \underbrace{\gamma \times \mathbf{P}_i}_{\text{Auto-PE}} . \tag{4.6}$$

Why is Auto-PE necessary? Could an APE-like method with learnable values not just learn to decrease the norm of the PE by itself? We posit that it could, but that Auto-PE learns this solution much more effectively, because it is a classic example of an effective inductive bias: it captures the modulation of the PE norm in a single parameter, making it easier for the model to learn the modulation.

Auto-PE can be applied to any APE-based method that adds the position embedding to the patch embedding, by applying the gating to the computed position embedding. We initialize $\gamma = 0.5$ and optimize γ end-to-end with a learning rate of 1.0×10^{-1}. We tuned these hyperparameters on the test sets of EuroSAT and Flowers-102.

Auto-RoPE. We design a special case of Auto-PE for RoPE [17] by applying γ to the learned angles used in the complex-valued multiplication (i.e. rotation) that RoPE applies in the attention operation. In this way, when γ approaches zero, the angles approach zero and rotation is reduced, reducing positional information. We learn a single parameter γ for each attention operator in the network. See Section 4.B for details.

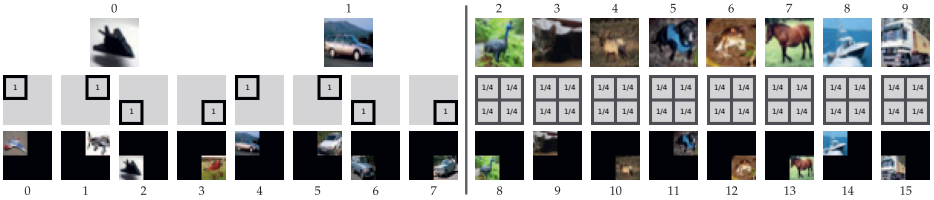


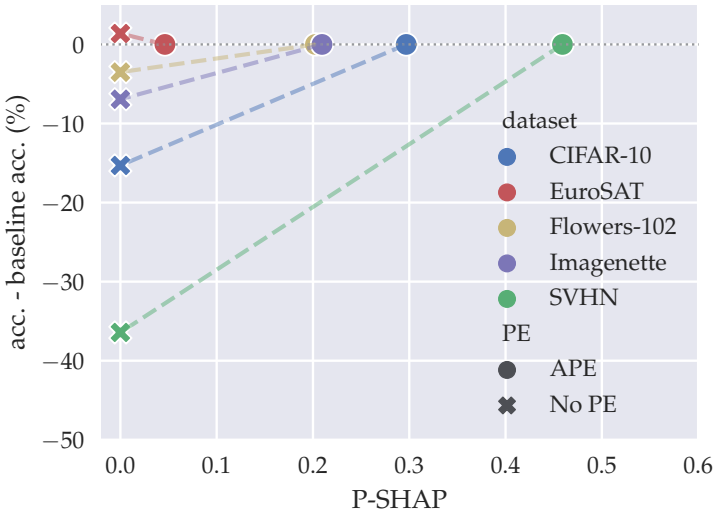
Figure 4.2: CIFAR-10-Position toy dataset construction. Top row shows samples from CIFAR-10, middle row shows the possible positions and their probabilities for each class in CIFAR-10-Position, and bottom row shows samples from all classes in CIFAR-10-Position.

4.5 Experiments

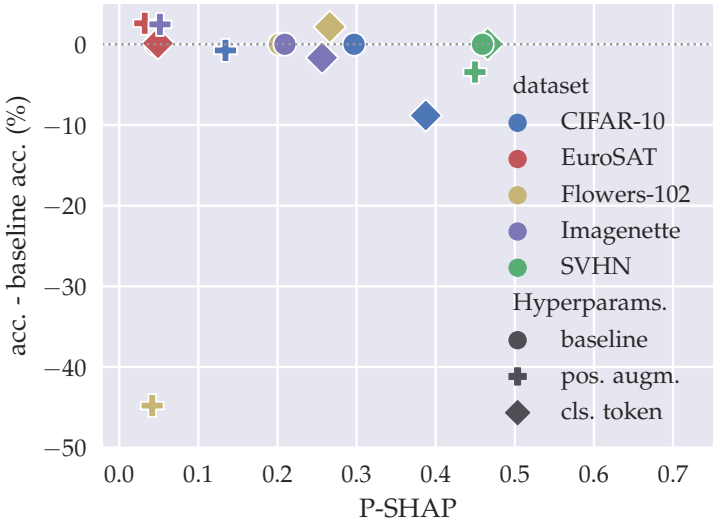
4.5.1 Controlled setting

We create a controlled setting to confirm that samples with known position bias score higher P-SHAP than samples with known lack of position bias. To this end we introduce the dataset *CIFAR-10-Position* (see Fig. 4.2). We rescale each CIFAR-10 sample to 16×16 pixels and place it in a corner of a 32×32 pixel frame. For classes that we designate to depend on position, we always place the image in the same corner. For classes independent of position, we place the image in a random corner. CIFAR-10-Position contains eight position-dependent classes with samples X_P and eight position-independent classes with samples X_{NP}. CIFAR-10-Position uses the same training and test split as the original CIFAR-10 dataset.

Our hypothesis is that the samples $\mathbf{X}^{\text{P+}}$ of the position-dependent classes have a higher P-SHAP score than the samples $\mathbf{X}^{\text{P-}}$ of the position-independent classes. To test this, we train a Vision Transformer from scratch on CIFAR-10-Position and measure P-SHAP over the test set. We use a non-parametric Mann-Whitney U-test [38] with the null hypothesis that P-SHAP($\mathbf{X}^{\text{P+}}$) = P-SHAP($\mathbf{X}^{\text{P-}}$) and the alternative hypothesis P-SHAP($\mathbf{X}^{\text{P+}}$) > P-SHAP($\mathbf{X}^{\text{P-}}$). We find that the p-value of this test is below 1%, indicating that P-SHAP can identify position-dependent samples in a model with mixed position bias. Section 4.D analyses the distribution of P-SHAP values within a dataset.



(a) Sec. 4.5.2: classification datasets



(b) Sec. 4.5.3: model hyperparameters

Figure 4.3: Models trained from scratch on several classification datasets. Dashed lines indicate best linear fit per dataset. (a) The higher the P-SHAP in the APE model, the more performance is lost by removing the PE. (b) Position-affecting data augmentations reduce P-SHAP with predictable effects on performance, while using a class token increases P-SHAP with varying effects on performance.

4.5.2 P-SHAP ON CLASSIFICATION DATASETS

To confirm that P-SHAP measures position bias in models trained on real-world datasets, we measure P-SHAP on Vision Transformers trained on two datasets chosen for their known position bias (EuroSAT and SVHN) as well as three datasets with hypothesized capture bias (CIFAR-10, Oxford-Flowers-102 and Imagenette). The training setting is detailed in Section 4.G. Notably, we use a modified version of RandAugment that does not apply any position-affecting augmentations, and we use global average pooling instead of class tokens, to benefit clean experimentation w.r.t. position bias.

Low position bias. EuroSAT [14, 15] (Fig. 4.1) is an image classification dataset of satellite imagery. The classification label pertains to a feature of the landscape in the picture, *e.g.*a river or a highway. The imagery is translation invariant because the dataset lacks capture bias. Therefore, there is no position bias in the dataset.

High position bias. SVHN [13] (Fig. 4.1) is an image classification dataset consisting of pictures of house numbers. While house numbers often consist of multiple digits, the images in this dataset are centered on a single digit to be classified. Therefore the network needs to distinguish between middle and other digits to learn to perform the task. Therefore, this dataset has high position bias.

Medium position bias: capture bias. CIFAR-10 [39], Oxford-Flowers-102 [40] and Imagenette [41], a ten-class subset of Imagenet [42] (Fig. 4.1), all consist of natural images taken by photographers and therefore suffer from capture bias. This means that in some samples the object of interest will be in center of the image. We therefore hypothesize that P-SHAP will measure medium levels of position bias, i.e. between the levels measured for EuroSAT and SVHN.

Results. Fig. 4.3a shows P-SHAP measured over the test set against the gain in test set accuracy compared to the baseline (APE) model. We find our hypotheses about the level of position bias are confirmed: we find that EuroSAT benefits from low P-SHAP, while SVHN benefits from high P-SHAP. Notably, EuroSAT performs better *without position embeddings*, while SVHN performs much worse without position embeddings. For the capture bias datasets, we find that all three datasets learn medium-high P-SHAP, compared to the datasets with known position bias. Correspondingly, we find that these datasets too lose performance when removing the PE, though less than for the high position bias dataset. These performance patterns match the known position bias of the datasets, showing that position bias plays a crucial role in Vision Transformers.

4.5.3 Effects of model hyperparameters

Position-affecting data augmentations. All our models are trained without data augmentations that apply spatial transformations to the data. In this experiment, we compare models trained *with* such augmentations to models without. Fig. 4.3b shows that position-affecting augmentations reduce the learned P-SHAP, and that datasets with medium to high P-SHAP lose performance with position-affecting augmentations, while datasets with lower P-SHAP gain performance. Notably, the Oxford-Flowers-102 dataset seems to be an outlier, where position augmentation almost breaks the model entirely.

CLS token versus average pooling. All our models are trained with average pooling instead of using a class token. Fig. 4.3b shows that using a class token increases P-SHAP on some datasets, but performance changes unpredictably.

Transfer learning. We find pre-training checkpoints trained on ImageNet [42] contain the roughly the same level of position bias as models trained from scratch on Imagenette, implying that models trained on more data *do not* learn less position bias. See Section 4.H for more details and results.

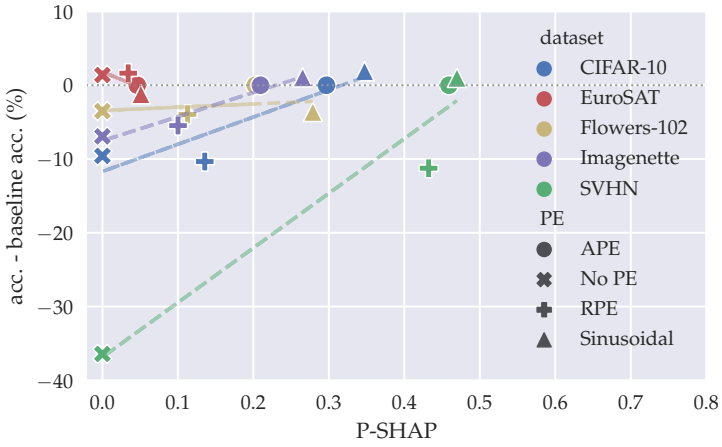
4.5.4 Tuning for position bias

Given the above findings, we optimize the models trained on datasets with known position bias using the settings that maximize alignment with the datasets position bias. Table 4.1 shows the settings used in training as well as the results. We find that using our recommended settings significantly improves performance on the tested datasets with low and high position bias, while results are mixed on datasets with medium position bias.

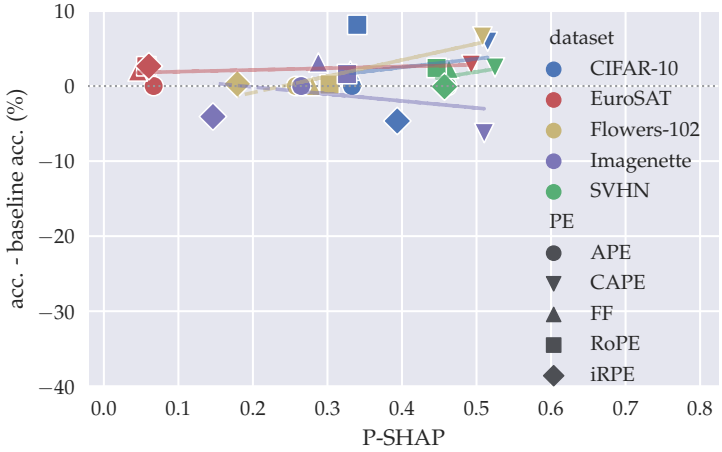
4.5.5 Evaluating position embeddings

We hypothesize that the choice of position embedding is important to tune the model to the position bias of the dataset. We compare the effects on P-SHAP of using a variety of position embedding methods. We evaluate both baseline PEs as well as original contribution PEs.

Results. Fig. 4.4 shows the results. For the baseline methods, a similar trend as seen in the above experiments is notable: the higher the position bias in the APE model, the better the performance for models that learn more position bias. For the original contribution methods, no single overarching pattern in the relationship between position bias and accuracy emerges. We discuss more subtle patterns in Section 4.F. Crucially, RoPE improves results on all datasets



(a) Section 4.5.5: baseline PE methods



(b) Section 4.5.5: original contrib. PE methods

Figure 4.4: Models with varying PEs trained from scratch on several classification datasets. Dashed lines indicate best linear fit per dataset. (a) For baseline PE methods, the higher the P-SHAP in the APE model, the more performance is lost by removing the PE. (b) For original contribution PE methods, there is no clear overarching trend.

Table 4.1: Models trained from scratch to optimize for the datasets position bias. Our "tuned" setting improves accuracy significantly on low and high position bias settings we test. Scores in bold are the best in their row with a margin of one standard deviation.

dataset	*default (APE)*	*tuned*	*settings of "tuned"*
EuroSAT	90.7 ± 0.2	**92.4 ± 0.5**	No PE, GAP, position-affecting augm.
CIFAR-10	76.5 ± 0.6	76.2 ± 0.2	APE, CLS token, no position-affecting augm.
Flowers-102	12.9 ± 1.4	**60.7 ± 2.4**	
Imagenette	**79.1 ± 1.2**	72.4 ± 1.4	
SVHN	90.9 ± 0.4	**94.8 ± 0.6**	APE, CLS token, no position-affecting augm.

RoPE may therefore obviate the need for a position bias-tuning PE method such as Auto-PE. However, in Section 4.5.6 we show that RoPE can be further improved using Auto-PE.

4.5.6 AUTO-PE

We evaluate our proposed Auto-PE on the same datasets using the same training settings as before (see Section 4.5.2). We apply Auto-PE to APE and RoPE, the best performing baseline and original contribution PE respectively from Section 4.5.5.

Results. Table 4.2 shows that, even though P-SHAP is only significantly affected for some models, Auto-PE matches or outperforms the best performance on all datasets. Notably, Auto-RoPE does not significantly affect P-SHAP in any model, but still outperforms RoPE in CIFAR-10 and SVHN. An explanation can be found in Fig. 4.I.2b in Section 4.I, which shows that Auto-PE enables RoPE to learn larger angles, leading to more discrimination between relative positions in RoPE attention. Section 4.I also contains additional results and analysis

4.6 CONCLUSION

In this work, we use our proposed Position-SHAP measure to show that position bias plays an important role in tuning Vision Transformers to good accuracy in image classification. We find that the amount of position bias present in the training dataset determines the optimal choice of position embedding, and propose Auto-PE, which obviates the need for choosing a position embedding

Table 4.2: Models trained from scratch with and without Auto-PE. Adding Auto-PE to either APE or RoPE equals or improves best performance on all datasets.

	APE		*RoPE*	
dataset	**APE**	**Auto-APE (ours)**	**RoPE**	**Auto-RoPE (ours)**
EuroSAT	90.7 ± 0.2	91.8 ± 0.3	**93.4 ± 0.6**	**93.5 ± 0.3**
CIFAR-10	77.4 ± 0.7	77.2 ± 2.6	85.5 ± 0.5	**86.3 ± 0.3**
Flowers-102	**58.6 ± 1.4**	**59.2 ± 1.0**	**58.8 ± 0.6**	57.7 ± 0.7
Imagenette	75.7 ± 0.7	**76.5 ± 0.5**	**77.3 ± 2.2**	**77.7 ± 1.3**
SVHN	94.9 ± 0.4	94.3 ± 0.1	97.3 ± 0.1	**97.6 ± 0.1**

method.

Limitations. Position-SHAP extends Kernel SHAP, which is very slow on models as small as the smallest Vision Transformers (see Section 4.E). In practice, hyperparameter tuning the position embedding is therefore unfortunately still more efficient than using P-SHAP. We do however see an opportunity to use DeepLIFT [43] as the backbone of P-SHAP, which should make computing P-SHAP more efficient than training multiple models. Unfortunately, there is currently no implementation of DeepLIFT that is compatible with Vision Transformers.

Future work. P-SHAP could be used to compare pre-trained checkpoints for how appropriate they are for a specific downstream dataset, or even as backbone network for another computer vision task. We hypothesize that backbones with high learned position bias may perform better than backbones with less position bias on tasks that rely on position information, such as object detection or instance segmentation. We also anticipate that P-SHAP could be useful as a sanity check for dataset creators, ensuring that the intended position bias is indeed present, or that no position bias is accidentally infused into the data.

REFERENCES

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929.

[2] V. Biscione and J. S. Bowers. *Convolutional Neural Networks Are Not Invariant to Translation, but They Can Learn to Be*. 2021. arXiv: 2110.05861 [cs.CV]. URL: https://arxiv.org/abs/2110.05861.

[3] O. S. Kayhan and J. C. v. Gemert. "On translation invariance in cnns: Convolutional layers can exploit absolute spatial location". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 14274–14285.

[4] R. Zhang. *Making Convolutional Networks Shift-Invariant Again*. 2019. arXiv: 1904.11486 [cs.CV]. URL: https://arxiv.org/abs/1904.11486.

[5] M. A. Islam, M. Kowal, S. Jia, K. G. Derpanis, and N. D. B. Bruce. *Global Pooling, More than Meets the Eye: Position Information is Encoded Channel-Wise in CNNs*. 2021. arXiv: 2108.07884 [cs.CV]. URL: https://arxiv.org/abs/2108.07884.

[6] I. B. Akkaya, S. S. Kathiresan, E. Arani, and B. Zonooz. *Enhancing Performance of Vision Transformers on Small Datasets through Local Inductive Bias Incorporation*. 2023. arXiv: 2305.08551 [cs.CV]. URL: https://arxiv.org/abs/2305.08551.

[7] J.-B. Cordonnier, A. Loukas, and M. Jaggi. *On the Relationship between Self-Attention and Convolutional Layers*. 2020. arXiv: 1911.03584 [cs.LG]. URL: https://arxiv.org/abs/1911.03584.

[8] S. d'Ascoli, H. Touvron, M. L. Leavitt, A. S. Morcos, G. Biroli, and L. Sagun. "ConViT: improving vision transformers with soft convolutional inductive biases*". In: *Journal of Statistical Mechanics: Theory and Experiment* 2022.11 (Nov. 2022), p. 114005. ISSN: 1742-5468. DOI: 10.1088/1742-5468/ac9830. URL: http://dx.doi.org/10.1088/1742-5468/ac9830.

[9] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy. "Do Vision Transformers See Like Convolutional Neural Networks?" In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 12116–12128. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/652cf38361a209088302ba2b8b7f51e0-Paper.pdf.

[10] B. Ren, Y. Liu, Y. Song, W. Bi, R. Cucchiara, N. Sebe, and W. Wang. "Masked Jigsaw Puzzle: A Versatile Position Embedding for Vision Transformers". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2023, pp. 20382–20391.

[11] S. Fabbrizzi, S. Papadopoulos, E. Ntoutsi, and I. Kompatsiaris. "A survey on bias in visual datasets". In: *Computer Vision and Image Understanding* 223 (2022), p. 103552. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2022.103552. URL: https://www.sciencedirect.com/science/article/pii/S1077314222001308.

[12] A. Torralba and A. A. Efros. "Unbiased look at dataset bias". In: *CVPR 2011*. 2011, pp. 1521–1528. DOI: 10.1109/CVPR.2011.5995347.

[13] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. "Reading Digits in Natural Images with Unsupervised Feature Learning". In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*. 2011. URL: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

[14] P. Helber, B. Bischke, A. Dengel, and D. Borth. "Introducing EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification". In: *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*. IEEE. 2018, pp. 204–207.

[15] P. Helber, B. Bischke, A. Dengel, and D. Borth. "Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (2019).

[16] E. D. Cubuk, B. Zoph, J. Shlens, and Q. Le. "RandAugment: Practical Automated Data Augmentation with a Reduced Search Space". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 18613–18624. URL: https://proceedings.neurips.cc/paper/2020/file/d85b63ef0ccb114d0a3bb7b7d808028f-Paper.pdf.

[17] B. Heo, S. Park, D. Han, and S. Yun. *Rotary Position Embedding for Vision Transformer*. 2024. arXiv: 2403.13298 [cs.CV]. URL: https://arxiv.org/abs/2403.13298.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[19] P. Shaw, J. Uszkoreit, and A. Vaswani. *Self-Attention with Relative Position Representations*. 2018. arXiv: 1803.02155 [cs.CL]. URL: https://arxiv.org/abs/1803.02155.

[20] Y. Li, S. Si, G. Li, C.-J. Hsieh, and S. Bengio. *Learnable Fourier Features for Multi-Dimensional Spatial Positional Encoding*. 2021. arXiv: 2106.02795 [cs.LG]. URL: https://arxiv.org/abs/2106.02795.

[21] T. Likhomanenko, Q. Xu, G. Synnaeve, R. Collobert, and A. Rogozhnikov. *CAPE: Encoding Relative Positions with Continuous Augmented Positional Embeddings*. 2021. arXiv: 2106.03143 [cs.LG]. URL: https://arxiv.org/abs/2106.03143.

[22] K. Wu, H. Peng, M. Chen, J. Fu, and H. Chao. *Rethinking and Improving Relative Position Encoding for Vision Transformer*. 2021. arXiv: 2107.14222 [cs.CV]. URL: https://arxiv.org/abs/2107.14222.

[23] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu. *RoFormer: Enhanced Transformer with Rotary Position Embedding*. 2023. arXiv: 2104.09864 [cs.CL]. URL: https://arxiv.org/abs/2104.09864.

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[25] R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, and J. Yosinski. *An Intriguing Failing of Convolutional Neural Networks and the CoordConv Solution*. 2018. arXiv: 1807.03247 [cs.CV]. URL: https://arxiv.org/abs/1807.03247.

[26] R. A. Rojas-Gomez, T.-Y. Lim, M. N. Do, and R. A. Yeh. *Making Vision Transformers Truly Shift-Equivariant*. 2023. arXiv: 2305.16316 [cs.CV]. URL: https://arxiv.org/abs/2305.16316.

[27] N. Watters, L. Matthey, C. P. Burgess, and A. Lerchner. *Spatial Broadcast Decoder: A Simple Architecture for Learning Disentangled Representations in VAEs*. 2019. arXiv: 1901.07017 [cs.LG]. URL: https://arxiv.org/abs/1901.07017.

[28] S. Jelassi, M. E. Sander, and Y. Li. *Vision Transformers provably learn spatial structure*. 2022. arXiv: 2210.09221 [cs.CV]. URL: https://arxiv.org/abs/2210.09221.

[29] W. Yu, M. Luo, P. Zhou, C. Si, Y. Zhou, X. Wang, J. Feng, and S. Yan. *MetaFormer Is Actually What You Need for Vision*. 2022. arXiv: 2111.11418 [cs.CV]. URL: https://arxiv.org/abs/2111.11418.

[30] O. S. Kayhan and J. C. v. Gemert. "On Translation Invariance in CNNs: Convolutional Layers Can Exploit Absolute Spatial Location". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[31] Z. Kong, P. Dong, X. Ma, X. Meng, M. Sun, W. Niu, X. Shen, G. Yuan, B. Ren, M. Qin, H. Tang, and Y. Wang. *SPViT: Enabling Faster Vision Transformers via Soft Token Pruning*. 2022. arXiv: 2112.13890 [cs.CV]. URL: https://arxiv.org/abs/2112.13890.

[32] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. 2021. arXiv: 2103.14030 [cs.CV]. URL: https://arxiv.org/abs/2103.14030.

[33] P. Ding, D. Soselia, T. Armstrong, J. Su, and F. Huang. *Reviving Shift Equivariance in Vision Transformers*. 2023. arXiv: 2306.07470 [cs.CV]. URL: https://arxiv.org/abs/2306.07470.

[34] M. Caron, N. Houlsby, and C. Schmid. *Location-Aware Self-Supervised Transformers for Semantic Segmentation*. 2023. arXiv: 2212.02400 [cs.CV]. URL: https://arxiv.org/abs/2212.02400.

[35] L. B. Guglielmo Camporese Elena Izzo. "Where are my Neighbors? Exploiting Patches Relations in Self-Supervised Vision Transformer". In: *British Machine Vision Conference (BMVC)*. 2022.

[36] Y. Liu, E. Sangineto, W. Bi, N. Sebe, B. Lepri, and M. De Nadai. "Efficient Training of Visual Transformers with Small Datasets". In: *Conference on Neural Information Processing Systems (NeurIPS)*. 2021.

[37] S. M. Lundberg and S.-I. Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

[38] H. B. Mann and D. R. Whitney. "On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other". In: *The Annals of Mathematical Statistics* 18.1 (1947), pp. 50–60. DOI: 10.1214/aoms/1177730491. URL: https://doi.org/10.1214/aoms/1177730491.

[39] A. Krizhevsky, G. Hinton, *et al. Learning multiple layers of features from tiny images*. Tech. rep. 2009.

[40] M.-E. Nilsback and A. Zisserman. "Automated Flower Classification over a Large Number of Classes". In: *Indian Conference on Computer Vision, Graphics and Image Processing*. Dec. 2008.

[41] *GitHub - fastai/imagenette: A smaller subset of 10 easily classified classes from Imagenet, and a little more French*. https://github.com/fastai/imagenette. Accessed: 2025-04-29.

[42] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[43] A. Shrikumar, P. Greenside, and A. Kundaje. *Learning Important Features Through Propagating Activation Differences*. 2019. arXiv: 1704.02685 [cs.CV]. URL: https://arxiv.org/abs/1704.02685.

APPENDICES

4.A ESTIMATING KERNEL SHAP VALUES

To estimate SHAP values $\boldsymbol{\phi}^{\mathbf{X}}$ for each sample $\mathbf{X}$, an weighted linear regression problem is solved. The samples in this problem are copies of $\mathbf{X}$, each with one combination of features removed from the sample. However, since neural networks do not support samples with removed features, removal of a feature is approximated by replacing it with a *background value*: the mean value of the feature in a *background dataset*, which should have the same distribution as the sample, *e.g.*the training dataset. The loss function optimized in the linear regression problem minimizes the difference between the network response $f(\mathbf{X}')$ on the modified sample $\mathbf{X}'$ and $g(\mathbf{X}')$, as shown in Theorem 2 of Lundberg and Lee [1], reproduced here in simplified form:

$$\pi(\mathbf{X}') = \frac{(M-1)}{(M \; choose \; |\mathbf{X}'|)|\mathbf{X}'|(M-|\mathbf{X}'|)} \tag{4.7}$$

$$L(f,g,\pi) = \sum_{\mathbf{X}' \in \text{RemoveFeatures}(\mathbf{X})} \left[f(\mathbf{X}') - g(\mathbf{X}')\right]^2 \pi(\mathbf{X}'), \tag{4.8}$$

$$\tag{4.9}$$

where $\mathbf{X} \in \mathbb{R}^{HWC}$, RemoveFeatures() creates one copy of $\mathbf{X}$ for all possible combinations of features and replaces those features with the background values, and $|\mathbf{X}'|$ is the number of non-removed features in $\mathbf{X}'$. For more details, we refer to Lundberg and Lee [1].

4.B AUTO-ROPE

RoPE. RoPE [2, 3] is a relative position embedding method for Vision Transformers. It rotates the output of the query-key product of the attention operator in the complex domain, using (predefined but end-to-end finetuned) angles θ multiplied by the relative distance m between the query and key token:

$$g(\boldsymbol{x}_m, \boldsymbol{x}_n, m-n) = \mathrm{Re}[(\boldsymbol{W}_q \boldsymbol{x}_m)(\boldsymbol{W}_k \boldsymbol{x}_n)^* e^{i(m-n)\theta}] \tag{4.10}$$

where g computes RoPE attention for input tokens $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$, that are $m-n$ positions apart, by multiplying the query-key product with complex-valued $e^{i(m-n)\theta}$, which amounts to rotation in the complex domain by a factor dependent on (learnable) parameter θ and the relative position difference $m-n$. This is the computation of RoPE for a single combination of query and key tokens. For more details, see Suo *et al*. [2] for the original RoPE formulation and Heo *et al*. [3] for the Vision Transformer formulation.

The angles θ are initialized as they are in the sinusoidal APE encoding [4], but they are fine-tuned in an end-to-end manner during training. This fine-tuning allows the model to "unlearn" the relative position embedding that RoPE applies, by learning $\theta = 0$. The results in Section 4.5.5 suggest that RoPE is reasonably successful in this. However, we proceed to implement Auto-RoPE and show in Section 4.5.6 that further improvement is possible.

Auto-RoPE. We design a special case of Auto-PE for RoPE [3], implementing the position embedding modulation of Auto-PE in RoPE by applying a learned weight γ to θ:

$$f_q(\boldsymbol{x}_m, m) = (\boldsymbol{W}_q \boldsymbol{x}_m) e^{im\gamma\theta} \tag{4.11}$$

$$f_k(\boldsymbol{x}_n, n) = (\boldsymbol{W}_k \boldsymbol{x}_n) e^{in\gamma\theta}. \tag{4.12}$$

$$\tag{4.13}$$

Auto-RoPE modulates the angle of rotation that RoPE applies. When $\gamma = 0$, the effective angle of rotation $\theta' = \gamma\theta = 0$. We learn a single parameter γ for each attention operator in the network.

4.C Validity of Kernel SHAP on ViT

Leemann *et al*. [5] argue that model linearity is an incorrect assumption for Vision Transformers (ViTs), and therefore Kernel SHAP should not be used on ViTs. Their theoretical argument is based on the assumption that identical tokens can exist in the input to the transformer model. This is indeed the case when using ViTs on text, as the same word appearing more than once in a sentence will be represented by identical tokens. However, because Vision Transformers take linearly projected patches as input tokens to which further position embeddings

are added, we argue the likelihood of encountering identical tokens is negligible. Additionally, the empirical results in Leemann *et al*. [5] suggest that Kernel SHAP performs well in non-synthetic datasets. We therefore argue that SHAP is a valid feature attribution method for Vision Transformers, and use it in this sense in our work.

4.D Analysis of P-SHAP values of individual samples

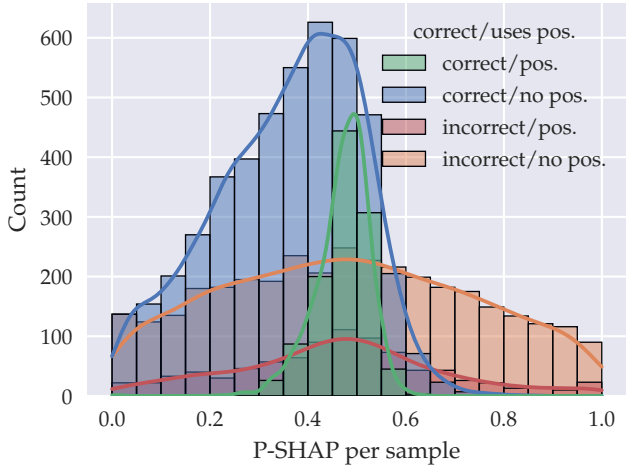


Figure 4.D.1: Histogram of CIFAR-10-Position validation samples, split by correct/incorrect prediction and position in- or dependent. Samples independent of position still use position embeddings to link neighboring patches, but P-SHAP is on average higher in position-dependent samples.

Fig. 4.D.1 shows a histogram of P-SHAP values of all samples in the test set for the experiment in Section 4.5.1, split by correctly and incorrectly classified samples. This shows the test passes because the means of the distributions of the correctly classified samples are clearly different, but still there is some overlap between the distributions. In any model, we will measure some position bias for every sample, since position embeddings are necessary to link neighboring image patches regardless of their position. Still, we measure more position bias for samples dependent on the position of the object of interest.

Furthermore, there is a clear difference between the P-SHAP values of correctly classified samples and incorrectly classified samples. To reduce noise in the P-SHAP measurement, we only measure P-SHAP for correctly classified samples.

4.E Effects of P-SHAP hyperparameters

P-SHAP variance. We run P-SHAP six times on the same trained model checkpoint to compute the standard deviation induced from the randomness in P-SHAP alone. This standard deviation is 0.0039 for a model that scores a mean P-SHAP of 0.3854.

Batch size. We compute P-SHAP by taking half of the batch as background to compute the P-SHAP values of the other half of the batch, and vice versa. The size of the batch therefore has an effect on the P-SHAP estimate, as it determines the size of the background dataset, which will be a better estimate of the true background with more samples. In our experiments the default batch size is 32 samples. We conduct an experiment finding that a larger batch size increases P-SHAP inference time notably without substantial effect on the variance in the P-SHAP estimate.

Inference cost. P-SHAP is on average 5571 times slower than regular validation inference: on CIFAR-10-Position, running the model once on the validation set takes approx. eight seconds, while running P-SHAP takes on average 11 hours and 12 minutes. We address this limitation further in Section 4.6.

4.F Analysis of P-SHAP on advanced PE methods

We analyze the results of Section 4.5.5 in more detail. See Fig. 4.F.1 for a large reproduction of Fig. 4.4b.

CAPE increases P-SHAP the most of all advanced PE methods, with strongly varying effect on accuracy: it performs worse on ImageNette, but (on par with) best on the other datasets. FF and RoPE do not increase P-SHAP significantly, but do increase performance on almost all datasets. This indicates the inductive bias of a PE can contribute to better performance in a different way than merely increasing the learned position bias. Finally, iRPE at best does not improve performance, and at worse reduces performance by a large margin. We do note

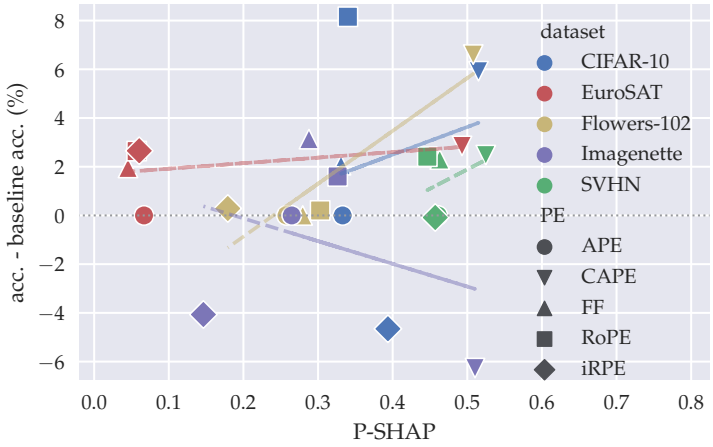


Figure 4.F.1: Reproduction of Fig. 4.4b in Section 4.5.5: original contribution PE methods

that on EuroSAT, the dataset with the lowest position bias, the earlier observed pattern that learning more position bias does not help, can be noted here as well. Similarly, on SVHN, the dataset with the highest position bias, all methods learn high position bias, though the slight performance differences we see cannot be directly attributed to higher learned position bias. We conclude that the benefits of advanced PE methods are not mainly in how they increase position bias, but rather may be in the way in which they allow models to process positional information.

4.G Training details

Default setting (Sections 4.5.2 to 4.5.6). We adapt the setup of Gani *et al*. [6]: we use ViT models [7] with 4px patch size, 192 dimensions in the hidden layer, 384 dimensions in the MLP, 12 heads and 9 layers, applying dropout with magnitude 0.1, training from scratch while downsampling all datasets to 32px image resolution. For data augmentations, we MixUp [8], CutMix [9], random erasing [10] and RandAugment [11]. However, we exclude RandAugment operations that affect the image spatially (e.g. rotation, scaling, shearing, etc.), as we perform a separate experiment in Section 4.5.3 on to test the effects of such data augmentations. We train all models for 400 epochs with batch size 512, using an Adam optimizer [12] with learning rate 1e−3 decayed using cosine

annealing [13], and label smoothing with magnitude 0.1. The models using APE position embedding are treated as the default models in our experiments, and perform comparable to published results with similar model sizes [7]. We train each model three times and average the results.

Transfer learning setting (Section 4.H). We use the same setup as in Section 4.5.2, except we change the model to use the original ViT implementation [7] of the `ViT-Tiny/16` model, with pretrained weights as published on Huggingface [14][1]. This model is larger than the model we used in previous experiments, but the smallest ViT model that we could find and successfully fits to the downstream datasets, even training from scratch. We train each model only once, to save compute.

4.H Effects of transfer learning

For transfer learning to be successful, the pre-trained model checkpoints should either be free of any dataset-specific bias or this bias needs to match the downstream dataset. We test if P-SHAP is learned during pre-training by measuring P-SHAP in the checkpoint. We then compare P-SHAP in models trained from scratch and with transfer learning on the aforementioned datasets. The training setting is detailed in Section 4.G.

Position bias in pre-trained model. We compute P-SHAP for the checkpoint on a random subset of 43.1% of the ImageNet validation set. Fig. 4.H.1 shows that this model scores a P-SHAP of ≈ 0.2, which we consider to be a medium-low amount of position bias. Interestingly, this score is not zero or even as low as learned on the low position bias dataset EuroSAT, implying that ImageNet pre-training does learn some position bias that is passed on to the downstream models. Furthermore, the measured position bias roughly matches the measured position bias of the related dataset Imagenette, which has only ten classes and $100\times$ less samples. This implies that models trained on more data *do not* learn less position bias.

Accuracy in downstream models. As expected, models using transfer learning perform significantly better than models trained from scratch. Interestingly, the higher the position bias, the bigger the gap in accuracy between fine-tuning and training from scratch. This could indicate that position bias actually hinders models trained from scratch more than those that are fine-tuned. We do note that CIFAR-10 seems to be a slight outlier in this trend.

[1] Weights from https://huggingface.co/WinKawaks/vit-tiny-patch16-224/tree/main

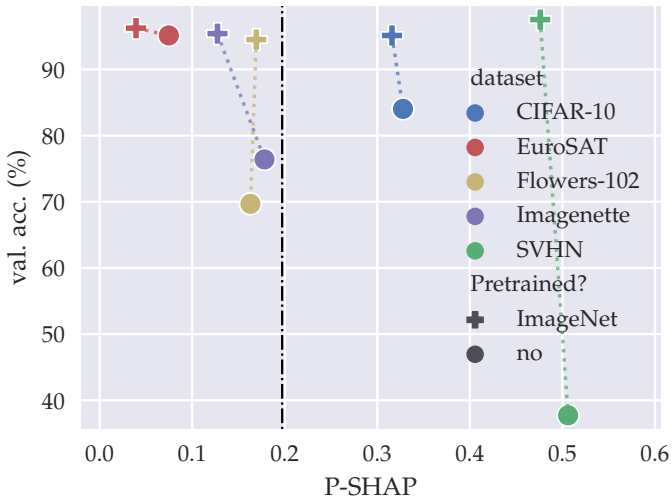


Figure 4.H.1: Results for models trained with and without pre-trained weights. Dash-dot line indicates the P-SHAP of the pre-trained checkpoint. We find that transfer learning affects position bias, slightly reducing it. Datasets with high position bias lose more accuracy by leaving out fine-tuning than datasets with low position bias.

Position bias in downstream models. Notably, position bias seems to be slightly smaller in fine-tuned models than in models trained from scratch. We speculate this could be because of the model needing to re-adjust its learned position bias to the position bias of the downstream dataset, which may be difficult, and lead to the model not being able to use position bias as well.

4.I Analysis of Auto-PE results

?? shows the results of Auto-PE plotted against position bias. Auto-PE does not seem to significantly alter position bias in any consistent manner, compared to models trained without Auto-PE. One exception is the Auto-APE model on EuroSAT, which we know from Section 4.5.2 needs to unlearn the PE. This model learns less position bias and improved accuracy, like when using no PE.

Fig. 4.I.2a shows the absolute value of learned γ parameters in Auto-PE. We note that learned values of γ are sometimes negative, thereby inverting the values

Table 4.I.1: Combining the tuning for position bias with Auto-PE: models trained from scratch with and without Auto-PE, on the baseline and "tuned" setting from Section 4.5.4. Combining our recommended "tuned" setting with Auto-PE does not lead to further improved results. Scores in bold are the best in their row with a margin of one standard deviation.

	standard setting		*+ Auto-PE*		*+ tuned*		*+ tuned + Auto-PE*	
dataset	**APE**	**RoPE**	**Auto-APE**	**Auto-RoPE**	**APE/No PE**	**RoPE**	**Auto-APE**	**Auto-RoPE**
EuroSAT	90.8 ± 0.3	92.2 ± 0.2	92.0 ± 0.7	92.3 ± 0.5	92.4 ± 0.6	**93.3 ± 0.4**	91.6 ± 0.2	93.5 ± 0.2
CIFAR-10	76.5 ± 0.6	**88.1 ± 0.3**	75.9 ± 0.2	86.5 ± 1.1	76.2 ± 0.2	83.5 ± 0.6	76.2 ± 1.1	84.3 ± 0.2
Flowers-102	12.9 ± 1.4	13.4 ± 0.6	14.0 ± 0.9	13.6 ± 0.5	**60.7 ± 2.4**	**58.7 ± 1.4**	**59.4 ± 0.1**	58.4 ± 1.9
Imagenette	79.1 ± 1.2	**81.7 ± 1.0**	79.5 ± 1.5	**80.9 ± 0.7**	72.4 ± 1.4	75.5 ± 1.3	73.1 ± 1.0	76.5 ± 0.3
SVHN	90.9 ± 0.4	93.4 ± 1.1	90.6 ± 0.3	86.3 ± 10.4	94.8 ± 0.7	**97.0 ± 0.3**	94.7 ± 0.4	**97.0 ± 0.2**

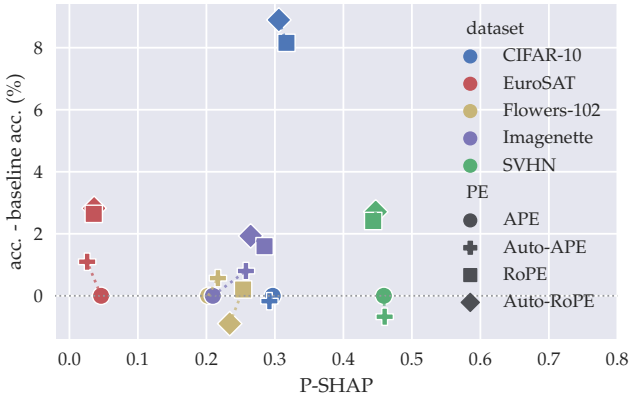
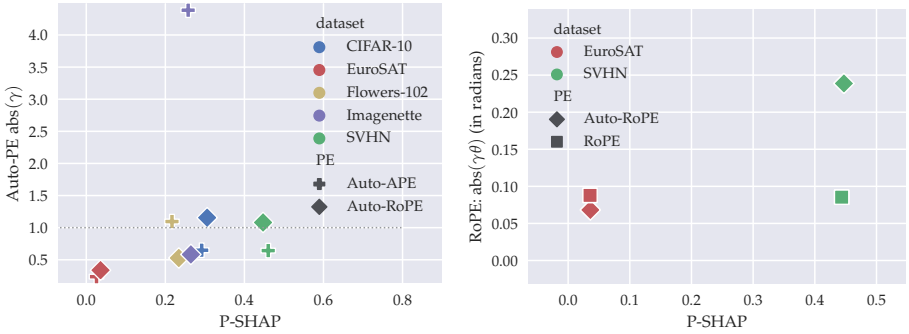


Figure 4.I.1: Results for Auto-PE. Dotted lines link models with and without Auto-PE. Auto-PE matches or improves the baseline PE in performance by learning the same or more dataset-appropriate position bias, as measured by P-SHAP.

of the PE. Since this should not have an effect on the position bias modeled, we plot the absolute value of the γ parameters. We note that Auto-PE nearly always learns to reduce the norm of the PE, though this effect is larger on datasets with low position bias.

Fig. 4.I.2b shows the effective rotation angles applied in Auto-RoPE attention, after the angles θ have been modulated by γ. Notably, on EuroSAT, Auto-RoPE reduces the effective angle, attenuating the position bias, while on SVHN the effective angle is increased, increasing the position bias as well.

Additional results. Preliminary experiments showed that Auto-RoPE with γ



(a) Sec. 4.5.6: Auto-PE: abs(γ)

(b) Sec. 4.5.6: Auto-RoPE learned angles

Figure 4.I.2: Learned values of (a) γ of Auto-APE and Auto-RoPE and (b) angles of Auto-RoPE. Dotted lines link models with and without Auto-PE. Auto-PE nearly always learns to reduce the norm of the PE, though this effect is larger on datasets with low position bias. In Auto-RoPE, Auto-PE allows RoPE's learned angles to converge to larger values, though we do not measure higher P-SHAP in these models. The γ-values for Auto-RoPE are the mean value of all γ parameters in the network.

shared between all attention operators performed slightly worse. We therefore did not consider this implementation in our work any further.

REFERENCES

[1] S. M. Lundberg and S.-I. Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

[2] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu. *RoFormer: Enhanced Transformer with Rotary Position Embedding*. 2023. arXiv: 2104.09864 [cs.CL]. URL: https://arxiv.org/abs/2104.09864.

[3] B. Heo, S. Park, D. Han, and S. Yun. *Rotary Position Embedding for Vision Transformer*. 2024. arXiv: 2403.13298 [cs.CV]. URL: https://arxiv.org/abs/2403.13298.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[5] T. Leemann, A. Fastowski, F. Pfeiffer, and G. Kasneci. "Attention Mechanisms Don't Learn Additive Models: Rethinking Feature Importance for Transformers". In: *Transactions on Machine Learning Research* (2025). ISSN: 2835-8856. URL: https://openreview.net/forum?id=yawWz4qWkF.

[6] H. Gani, M. Naseer, and M. Yaqub. *How to Train Vision Transformer on Small-scale Datasets?* 2022. arXiv: 2210.07240 [cs.CV]. URL: https://arxiv.org/abs/2210.07240.

[7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929.

[8] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. "mixup: Beyond Empirical Risk Minimization". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=r1Ddp1-Rb.

[9] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. "Cutmix: Regularization strategy to train strong classifiers with localizable features". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6023–6032.

[10] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang. "Random erasing data augmentation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 13001–13008.

[11] E. D. Cubuk, B. Zoph, J. Shlens, and Q. Le. "RandAugment: Practical Automated Data Augmentation with a Reduced Search Space". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 18613–18624. URL: https://proceedings.neurips.cc/paper/2020/file/d85b63ef0ccb114d0a3bb7b7d808028f-Paper.pdf.

[12] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

[13] I. Loshchilov and F. Hutter. *SGDR: Stochastic Gradient Descent with Warm Restarts*. 2017. arXiv: 1608.03983 [cs.LG]. URL: https://arxiv.org/abs/1608.03983.

[14] *Hugging Face – The AI community building the future.* https://huggingface.co/. Accessed: 2025-04-29.

5

Scale Learning in Scale-Equivariant Convolutional Networks

Objects can take up an arbitrary number of pixels in an image: objects come in different sizes, and, photographs of these objects may be taken at various distances to the camera. These pixel size variations are problematic for CNNs, causing them to learn separate filters for scaled variants of the same objects which prevents learning across scales. This is addressed by scale-equivariant approaches that share features across a set of pre-determined fixed internal scales. These works, however, give little information about how to best choose the internal scales when the underlying distribution of sizes, or scale distribution, in the dataset, is unknown. In this work we investigate learning the internal scales distribution in scale-equivariant CNNs, allowing them to adapt to unknown data scale distributions. We show that our method can learn the internal scales on various data scale distributions and can adapt the internal scales in current scale-equivariant approaches.

This chapter has been published as:
M. Basting, R. Bruintjes, T. Wiedemer, M. Kümmerer, M. Bethge, J.C. van Gemert. "Scale Learning in Scale-Equivariant Convolutional Networks". In: *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2024)*, 2024, pp. 567-574.

Code available at:
https://github.com/MBasting/Scale-Learning-in-Scale-Equivariant-Convolutional-Networks

5.1 Introduction

Objects in images naturally occur at various scales. The scale, or size in terms of pixels, of an object in an image can vary because of perspective effects stemming from the distance to the camera or due to interclass variation. For example, imagine a golf ball and a volleyball being classified as balls but varying in size. Vanilla CNNs can learn differently-sized objects when presented with large amounts of data. However, since the CNN has no internal notion of scale, separate filters for differently scaled versions of the same objects are learned, leading to significant redundancy in the learned features.

Scale-equivariant CNNs such as those proposed by Yuo *et al*. [1] and Sosnovik *et al*. [2] share features across a fixed set of chosen internal scales which increases parameter efficiency by removing the need to learn separate filters for differently-sized objects. Yet, such scale-convolution approaches need tune the internal scales as a hyper-parameter. Instead, here, we present a model that can learn these internal scales.

In this paper, we present a model of the relationship between the internal scales and the data scale distribution. We show empirically the parameters for which this model is most accurate. Furthermore, we define a parameterization of the internal scales and draw inspiration from NJet-Net [3] to learn the internal scales. Our method provides a way to learn the internal scales without the need for prior knowledge of the scale distribution of your data.

We have the following contributions:

1. We demonstrate that the best internal scales are related to the used data scale distribution.

2. We derive an empirical model that shows approximately how we should choose the internal scales when the data scale distribution is known.

3. We remove the need for prior knowledge about the data scale distribution by making the internal scales learnable.

5.2 Related Work

Scale spaces. Scale is naturally defined on a logarithmic axis [4, 5]. We base our work on Gaussian scale-space theory and use theory on the logarithmic nature of scale to define the internal scale tolerance model and in the parameterisation of the internal scales.

Pyramid Networks. Pyramid Networks use differently scaled versions of the input image to share features across different scales. Popular pyramid networks include methods proposed by Farabet *et al*. [6], Marcos *et al*. [7] and SI-ConvNets [8]. These methods are equivariant over fixed chosen scales and require many expensive interpolation operations. Contrarily, our approach can learn the scales without extensive use of interpolations.

Scale group convolutions. An alternative way to achieve scale-equivariance or scale-invariance is through the use of group convolution [1, 2, 9–12]. Sosnovik *et al*. [13] argue that the discretisation of the underlying continuous basis functions leads to increased scale-equivariance error and therefore leads to worse performance. Instead, they opt to use dilation for integer scale factors and directly optimise basis functions for non-integer scales using the scale-equivariance error. While all methods allow for non-integer scale factors, the scales over which the network is equivariant are fixed and they provide little instructions on how to best choose the internal scales.

Learnable Scale. Continuous kernel parameterisation forms the basis of methods that aim to learn the scale or scales of the dataset [3, 14–18]. The NJet-Net [3] learned the scale of the dataset by making the σ parameter of the Gaussian derivative basis function learnable. We build on that work to learn multiple internal scales simultaneously.

5.3 METHOD

In Fig. 5.1 we visualize the setting. Our method is equivariant to both translation and scale transformations. Like SESN [2], our method achieves scale-equivariance through an inverse mapping of the kernel:

$$L_s[f] * \kappa = L_s[f * \kappa_{s^{-1}}], \quad \forall f, s \tag{5.1}$$

where L_s represents a scaling transformation by a factor s, κ is a discretized continuous kernel parameterized by an inner scale. Thus, a scaled input convolved with a kernel is the same as first convolving the original input with an inversely scaled kernel and then applying the same scaling.

Due to the discrete nature of images, we need to approximate the equivariance to translation and scaling by a discrete group. The translation group is approximated by taking into account all discrete translations. The scaling group

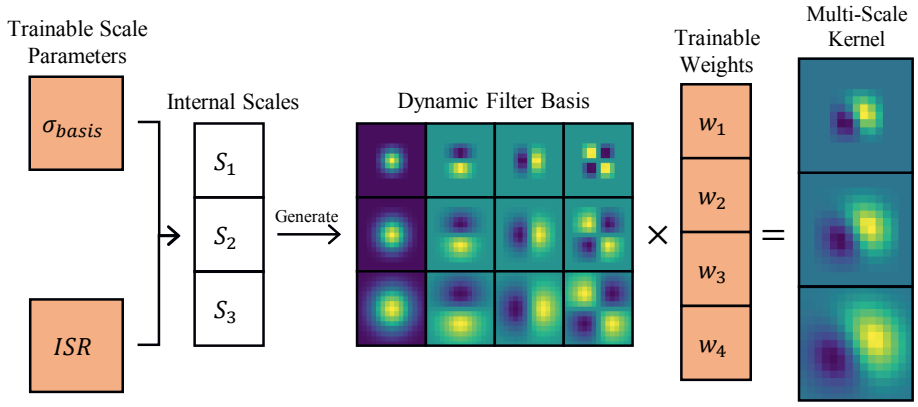


Figure 5.1: Dynamic Multi-Scale Kernel generation pipeline. Filter basis is parameterised by a discrete set of scales which in turn are generated from learnable parameters, controlling both the size of the first scale σ_{basis} and the range the internal scales span (ISR). Linear combination of the Dynamic Filter Basis functions with trainable weights form Multi-Scale Kernel.

is discretised by N_S scales with log-uniform spacing as follows:

$$S = [\sigma_{basis} \times ISR^{(\frac{i}{N_S - 1})} \text{ for } i \text{ in } 0..N_S - 1] \tag{5.2}$$

where σ_{basis} is a learnable parameter that defines the smallest scale, and the Internal Scale Range ISR defines the range between the largest and smallest scale. The logarithmic spacing can be attributed to the logarithmic nature of the scale.

The kernels of the model consist of a weighted sum of basis functions that are defined at each scale in the internal scales S. Following Sosnovik *et al*. [2], we use a basis of 2D Hermite polynomials with a 2D Gaussian envelope. This basis is pre-computed at the start of training for all pre-determined scales if scale learning is disabled. Otherwise, the basis functions are recomputed at each forward pass.

Scale-Convolution. Scale-convolution is a standard convolution extended by incorporating an additional scale dimension [2]. Without taking into account interscale interactions we define the following two types of scale convolutions:

1. Conv $T \rightarrow H$: In this scenario, the input of the scale-convolution is a tensor without any scale-dimension, or $|S'| = 1$. The output, defined over the internal scales S stems from the convolution of the input with scaled kernels $\kappa_{s^{-1}}$ s.t. $s \in S$:

$$[f *_H \kappa](s,t) = f(\cdot) * \kappa_{s^{-1}}(\cdot) \tag{5.3}$$

where κ_s is a kernel scaled by s, $*_H$ is the scale convolution and $*$ is a standard convolution.

2. Conv $H \rightarrow H$: The input is now defined over the internal scales S, the resulting output at scale s is the convolution of the input at scale s with the scaled kernel $\kappa_{s^{-1}}$:

$$[f *_H \kappa](s,t) = f(s,\cdot) * \kappa_{s^{-1}}(\cdot) \tag{5.4}$$

These methods are designed to adhere to the scale-equivariance equation highlighted in Eq. 5.1.

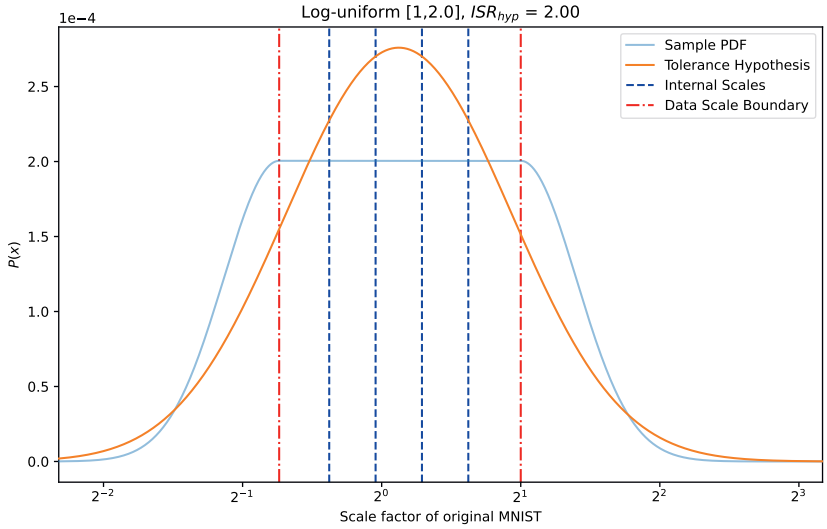


Figure 5.2: Example of possible internal scale tolerance model for the log-uniform data scale distribution over the range [0.6, 2.0] with ISR_{hyp} fixed to 2 to reflect internal scales choice in SESN [2] on MNIST-Scale.

Internal Scale Tolerance Model. We define an empirical tolerance model to estimate which internal scales to choose when the scale distribution is known. The tolerance describes the region of data scales the kernel can generalize to. Previous papers have shown that the generalization error to unseen scales follows an approximate log-normal distribution [8, 12]. Therefore, we use a normal distribution on a logarithmic scale to model the tolerance for each kernel at a certain internal scale. The log-normal distributions of each internal scale are then combined into one mixture model. An example of a possible configuration can be seen in Fig. 5.2.
The internal scale tolerance model has the following parameters:

1. Reference Internal Scale: defines the relationship between the position of the internal scales and the data scales.
2. ISR_{hyp}: range over which the internal scales are defined, this is the factor between the largest and the smallest scale.
3. τ_{tol}: standard deviation of the underlying log-normal distribution that is placed on each internal scale.

The reference scale and ISR are specific to each tolerance model of a data scale distribution while τ_{tol} is independent of the data scale distribution and a property of a kernel.

We make the assumption here that we do know the data scale distribution. We extend the data scale distribution at the boundaries by a half-log-normal distribution with $\sigma = 0.4$ to model the generalisation to unseen scales. The Kullback-Leibler divergence [19] is used to fit the tolerance model on the data scale sampling distributions.

Moving away from fixed Scale Groups. To make the scales of the network learnable we move away from fixed multi-scale basis functions and make the scales of the basis functions dynamic. The scales that parameterise the basis functions are continuous and have a gradient with regard to the loss allowing for direct optimisation. This allows us to simultaneously learn the kernel shape and scales, see Fig. 5.1. Unlike SEUNET [16], we do not parameterise the scales directly but parameterise the internal scales by σ_{basis} and the ISR using Eq. 5.5.

We observe that a value for the ISR lower or equal to 1 is unwanted as this would result in kernels at the same scale or a subsequent scale smaller than the base scale defined by σ_{basis}. We do not use a ReLU activation as this can lead to a dead neuron and zero gradient. We parameterise the ISR using the following formula:

$$ISR = 1 + \gamma^2 \tag{5.5}$$

where γ is the learnable parameter. We will only mention the ISR since it is closely related to the learnable parameter and more intuitive to understand.

Various size basis functions lead to difficulties in choosing the best kernel size before training. We use the method by Pintea *et al*. [3] to learn the size of the kernel based on the ISR:

$$l = 2\lceil k(\sigma_{basis} \times ISR)\rceil + 1 \tag{5.6}$$

where k is a hyperparameter that determines the extent of the approximation of the continuous basis functions. Thus, the kernel size used in convolution is determined by the largest scale in the set of internal scales which is directly parameterised by σ_{basis} and the ISR.

5.4 EXPERIMENTS

In Sections 5.4.1, 5.4.2, we use the simple architecture shown in Table 5.1 on the commonly used MNIST-Scale dataset [8] with a logarithmically uniform data scale distribution with a range of 1 to $2^{1.5}, 2^{2.25}$ and 2^3 corresponding to $2.83, 4.76,$ and 8 scale factors of MNIST respectively. Section 5.A gives a complete description of all datasets used in the experiments.

Table 5.1: CNN for Sections 5.4.1, 5.4.2 to show the impact of choosing the internal scales and scale-learning.

Conv $T \to H$, Hermite, $N_S = 3$, 16 filters
scale-projection
batch norm, relu
42 x 42 max pool
fully-connected, softmax

5.4.1 VALIDATION

Do internal scales really matter? We test our assumption that the Internal Scale Range (ISR), the factor between the largest and smallest internal scale, influences the performance. Furthermore, we compare the optimal ISRs we discovered with the suggested values of the Internal Scale Tolerance model. The ISRs are chosen on a logarithmic scale in the range of [1.5, 7.65].

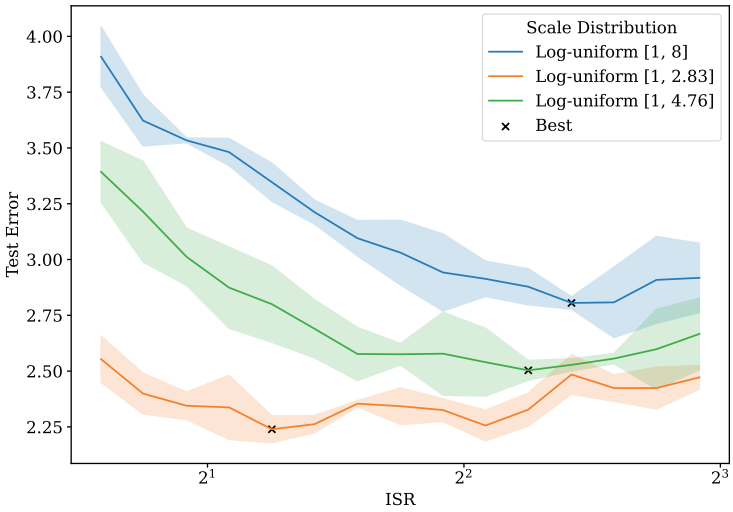


Figure 5.3: Impact on test error when varying Internal Scale Range (ISR) for different log-uniform data scale distributions from 1 scale factor of MNIST to 2.83, 4.76 and 8 scale factors MNIST. The value of the Internal Scale Range (ISR) for the best-performing model increases together with the width of the model.

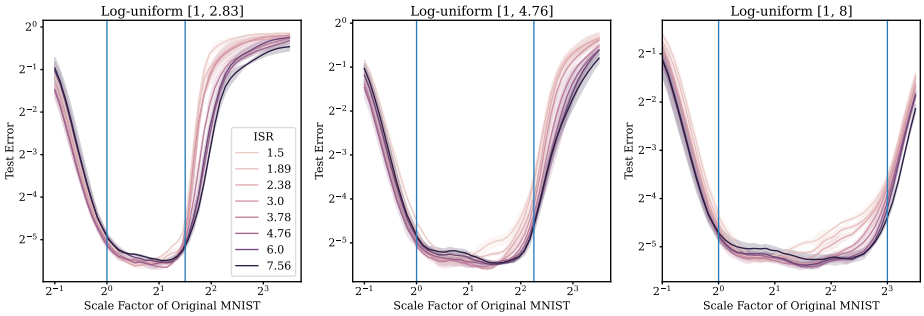


Figure 5.4: Test Error per data scale for multiple data scale distributions and values for the Internal Scale Range (ISR). Models with narrow internal scales especially deteriorate in performance in the large-scale region for wide distribution.

The results in Fig. 5.3 indicate that smaller ISRs are better for narrow data scale distributions, while larger ISRs perform better for wider data scale distributions. Thus, for a log-uniform distribution that spans a small scale range narrow internal scales are preferred. Conversely, for a log-uniform distribution that spans a large scale range, wide internal scales are preferred. Looking at the test error of individual data scales in Fig. 5.4, we see that at the boundary regions of wide scale distributions with narrow internal scales achieve significantly higher test error than wider internal scales, indicating that the model cannot share features across the whole data scale distribution. Narrow internal scales perform slightly better than wider internal scales when evaluated on narrow distributions. This statement aligns with results found in Fig. 5.3, which indicates less test error variation between ISR values for the log-uniform distribution between 1 and 2.83.

The results of the combined optimization of the tolerant model and the three sampling distributions can be found in Fig. 5.5. The optimization leads to $\tau_{tol} = 0.459$. The optimization fits ISR values approximately similar to the best ISRs depicted in Fig. 5.3. Again, the ISR values follow an increasing pattern when the data scale distribution gets wider. Additionally, Fig. 5.5 shows increasing gaps in the tolerance hypothesis between internal scales for wide distributions.

Can we learn the internal scales? To test our scale-learning capabilities, we evaluate our scale-learning on three log-uniform data scale distributions. The ISR is parameterised according to Eq. 5.5 and the scale parameters are initialised with $\sigma_{basis} = 2$ and $ISR = 3$. The results of learning the ISR and σ_{basis} compared to the best-performing ISRs without scale learning enabled can be found in Table 5.2. Apart from the narrowest scale distribution, the ISR values learned increase when enlarging the range of the data scale distribution. Our scale learning method gives comparable performance to the baselines while not using hyperparameter optimisation to determine the best ISR.

5.4.2 Model Choices

How does initialization of the scales affect scale learning behavior? We test the importance of the initialization of the internal scales by varying the starting values of σ_{basis} and ISR and report the classification error and variation in learned scale parameters. We vary the σ_{basis} between 1 and 4 and the ISR between 1.5 and 6 in a logarithmic fashion.

Table 5.3 show the results for the log-uniform distributions between $[1, 2.83]$, $[1, 4.76]$ and $[1, 8]$. The results indicate that the learned σ_{basis} and ISR values can

Table 5.2: Learned parameters for the Basis Min Scale σ_{basis} and Internal Scale Range (ISR) compared to the configuration of the best-performing model with fixed internal scales on different ranges of the log-uniform data scale distribution. Apart from the log-uniform distribution with boundaries [1,2.83], the learned scale parameters σ_{basis} and ISR follow a similar pattern as the manually found scale parameters. The test error of our scale-learning method is also comparable to the best-performing models with fixed scales.

Data Scale Distribution	Scale Learning	σ_{basis}	ISR	Test Error
Log-uniform [1,2.83]	✓	1.96 ± 0.081	3.390 ± 0.545	2.291 ± 0.067
	✗	2	2.34	2.239 ± 0.060
Log-uniform [1,4.76]	✓	2.001 ± 0.063	3.321 ± 0.024	2.510 ± 0.084
	✗	2	4.76	2.503 ± 0.045
Log-uniform [1, 8.00]	✓	1.943 ± 0.064	4.196 ± 0.159	2.872 ± 0.070
	✗	2	5.35	2.805 ± 0.028

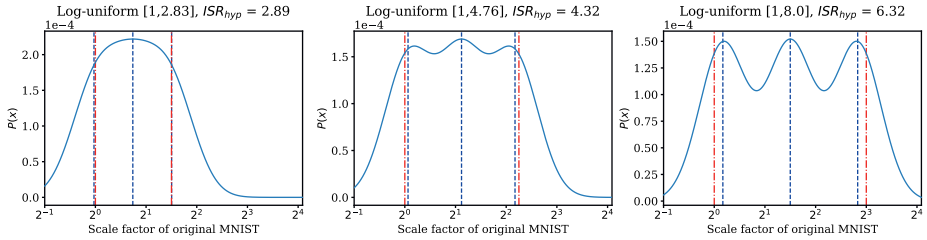


Figure 5.5: Results of combined optimization of tolerance hypothesis models and three log-uniform data scale distributions. The blue dashed line indicates the proposed internal scales, while the continuous blue line represents the tolerance hypothesis for a specific data scale distribution. The red dashed lines indicate the boundaries of the log-uniform distribution. The result of fitting the tolerance model results in increasing ISR_{hyp} similar to results best performing ISRs found for each data scale distribution in Fig. 5.3.

Table 5.3: Mean and standard deviation of learned scale parameters (σ_{basis}, ISR) and Test Error for different initialization of σ_{basis} and ISR for log-uniform data scale distribution between $[1, 2.83]$, $[1, 4.76]$ and $[1, 8]$. Learned σ_{basis} and ISR values are highly dependent on the values they are initialized on. Initialization with a large ISR for a wide data scale distribution leads to significantly lower test error than initialization at a low ISR.

Init σ_{basis}	Init ISR	Learned σ_{basis}	Learned ISR	Test Error
Data Scale Distribution: Log-uniform [1,2.83]				
1	1.5	1.268 ± 0.061	2.609 ± 0.269	2.487 ± 0.108
	3.0	1.236 ± 0.139	3.300 ± 0.082	2.357 ± 0.024
	6.0	1.313 ± 0.102	4.350 ± 0.396	2.309 ± 0.021
2	1.5	**1.810 ± 0.057**	**2.405 ± 0.167**	**2.260 ± 0.025**
	3.0	1.973 ± 0.079	3.635 ± 0.647	2.368 ± 0.055
	6.0	1.994 ± 0.012	5.336 ± 0.283	2.359 ± 0.097
4	1.5	2.778 ± 0.092	2.521 ± 0.199	2.421 ± 0.050
	3.0	2.703 ± 0.081	3.817 ± 0.245	2.483 ± 0.089
	6.0	2.832 ± 0.001	5.211 ± 0.416	2.420 ± 0.124
Data Scale Distribution: Log-uniform [1, 4.76]				
1	1.5	1.294 ± 0.110	3.462 ± 0.410	3.033 ± 0.130
	3.0	1.253 ± 0.070	3.829 ± 0.126	2.762 ± 0.087
	6.0	1.331 ± 0.091	4.612 ± 0.188	2.727 ± 0.101
2	1.5	1.931 ± 0.068	2.932 ± 0.169	2.767 ± 0.128
	3.0	1.975 ± 0.060	3.309 ± 0.092	2.527 ± 0.120
	6.0	**2.041 ± 0.092**	**4.882 ± 0.007**	**2.501 ± 0.157**
4	1.5	2.515 ± 0.038	3.093 ± 0.157	2.648 ± 0.073
	3.0	2.587 ± 0.040	3.638 ± 0.169	2.575 ± 0.070
	6.0	2.803 ± 0.098	5.662 ± 0.204	2.709 ± 0.078
Data Scale Distribution: Log-uniform [1, 8.00]				
1	1.5	1.450 ± 0.130	3.767 ± 0.171	3.087 ± 0.173
	3.0	1.275 ± 0.076	4.158 ± 0.123	3.131 ± 0.179
	6.0	1.331 ± 0.097	5.423 ± 0.529	3.087 ± 0.178
2	1.5	1.755 ± 0.156	3.444 ± 0.025	3.010 ± 0.198
	3.0	1.982 ± 0.068	4.095 ± 0.078	2.921 ± 0.082
	6.0	**2.079 ± 0.061**	**5.053 ± 0.375**	**2.745 ± 0.012**
4	1.5	2.453 ± 0.151	3.466 ± 0.216	2.935 ± 0.036
	3.0	2.607 ± 0.078	4.401 ± 0.343	2.803 ± 0.085
	6.0	2.655 ± 0.044	5.268 ± 1.071	2.957 ± 0.140

be adapted to fit the data scale distribution but the initialization of the values has a big impact on the learned scales and thus also the test error. Initialization with a large ISR for a wide data scale distribution leads to significantly lower test error. Fig. 5.6 shows the ISR over time during training, indicating the ISR stabilizes after around 20 epochs while the best-performing model has a significantly larger ISR.

How does parameterisation of learnable scales affect learnability? We test the importance of our scale learning parameterisation method on the stability and classification performance against other possible parameterisation methods. We initialise all scale learning approaches with the internal scales: $[2.0, 3.47, 6]$. The parameterisation methods that we compare are:

1. Learning the first scale (σ_{basis}) and the ISR using the parameterisation from Eq. 5.5 (Ours)
2. Learning the first scale (σ_{basis}) and the individual spacings between subsequent scales
3. Learning the individual scales directly, based on Yang *et al*. [16] but without defining intervals the internal scales adhere to.

Table 5.4 shows the parameterization methods, the classification error and the variation in the learned internal scales. Unlike shown in Yang *et al*. [16], directly learning the scales without constraints between the internal scales does not lead to internal scales converging to the same value. The methods do not vary significantly in performance for the log-uniform distribution between [1, 2.83] and [1, 4.76] but this changes when training on wider distributions. All methods adjust the scales somewhat to account for the wider scale distribution but our method of learning the Internal Scale Range (ISR) is more stable and achieves significantly better test error.

5.4.3 Comparing baselines

We compare our scale-learning ability against existing scale-equivariant baselines by evaluating on the MNIST-Scale [8] dataset. We reuse the code provided in DISCO [13] to compare our results to a baseline CNN and other methods that take into account scale variations such as SI-ConvNet [8], SS-CNN [9], SiCNN [20], SEVF [7], DSS [21], SESN [2] and DISCO [13]. All methods adopt the same training strategy, apart from our scale learning method having a different learning rate scheduler for its scale parameters.

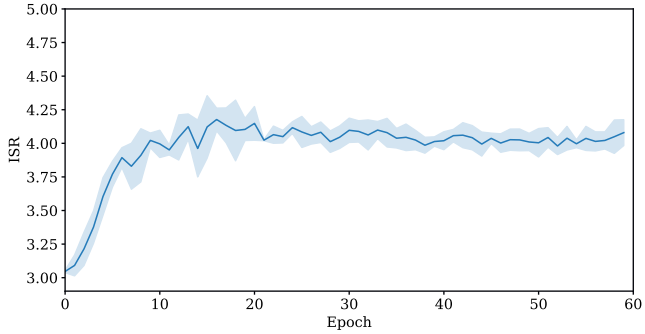


Figure 5.6: ISR parameter overtime for run initialized with $\sigma_{basis} = 2, ISR = 3$ on log-uniform distribution with boundaries [1,8]. After around 20 epochs, the learnable ISR stabilizes while the value for the ISR of the best-performing model is significantly larger.

Table 5.4: Mean and standard deviation of learned scales and Test Error of different parameterizations for multiple log-uniform distributions with internal scales initialized as [2.0, 3.46, 6.0]. Learning the ISR leads to more stable learned internal scales and better performance for wide distributions further away from the initialized scales.

Data Scale Distribution	Parameterization	Scale 1	Scale 2	Scale 3	Test Error
Log-uniform [1, 2.83]	Direct	1.965 ± 0.047	3.500 ± 0.193	6.235 ± 0.497	2.321 ± 0.095
	Individual Spacing	**1.967 ± 0.079**	**3.591 ± 0.329**	**6.930 ± 1.374**	**2.285 ± 0.038**
	ISR	1.960 ± 0.081	3.608 ± 0.435	6.672 ± 1.311	2.291 ± 0.067
Log-uniform [1, 4.76]	Direct	1.865 ± 0.046	3.357 ± 0.105	6.450 ± 0.049	2.554 ± 0.093
	Individual Spacing	1.996 ± 0.013	3.626 ± 0.158	6.830 ± 0.167	2.565 ± 0.061
	ISR	**2.001 ± 0.063**	**3.647 ± 0.127**	**6.647 ± 0.255**	**2.510 ± 0.084**
Log-uniform [1, 8.00]	Direct	1.689 ± 0.109	3.262 ± 0.107	6.997 ± 0.282	3.057 ± 0.015
	Individual Spacing	1.902 ± 0.085	3.648 ± 0.165	8.093 ± 0.229	3.007 ± 0.049
	ISR	**1.943 ± 0.063**	**3.977 ± 0.053**	**8.145 ± 0.057**	**2.872 ± 0.070**

We also compared our Internal Scale Range based parameterization with other parameterizations such as: learning the individual spacings between internal scales and learning the scales directly. Learning the scale directly is similar to the approach taken by Yang *et al*. [16] but without defining intervals the internal scales have to adhere to.

As can be seen from Table 5.5, the performance of the scale learning approaches are very comparable with SESN [2] without learnable scales. All three scale-learning approaches achieve test error performance within one standard deviation of SESN with fixed scales. The learned scales, found in Table 5.6, are consistently more spread out than the default scales used in SESN especially when scale data augmentation is used.

5.5 Discussion

We have shown to be able to learn the internal scales, but the problem of choosing the number of internal scales remains an issue. For wide scale distribution, wide internal scales achieve the best performance. However, if the models were truly scale-equivariant, the resulting test error would be similar to the test error for the log-uniform data scale distribution between 1 and 2.83. More specifically, if the spacing between the internal scales is too large the implied scale-equivariance over the entire range of the internal scales does not hold up. The model again needs to learn duplicate filters to cover the entire data scale range. This hypothesis also matches up with our Internal Scale tolerance model seen in Fig. 5.5, which shows dips in between internal scales. We expect that increasing the number of internal scales restores the scale equivariance over the entire scale group with the downside of reduced computational efficiency.

Another difficulty of learning the scales is the initialization of the internal scales. We have found that the initialization of the internal scales has a large impact on the learned scales and as a result the performance. However, we do expect that this can be resolved by tuning the training procedure.

In addition, our method adds a significant computational overhead since it has to reconstruct the dynamic filter basis functions on each step instead of being able to reuse the fixed multi-scale basis. However, hyperparameter optimization of the scale parameters would take significantly longer.

Learnable scales did not add significant gains for classification but for other tasks with larger scale-variations the importance of choosing internal scales becomes more important. We anticipate that the ability to learn the internal scales is especially beneficial in more complicated scenarios with more complicated data scale distributions, like a normal distribution. To learn the internal scales

Table 5.5: Classification error of Vanilla CNN and other methods that take into account scale variations in the data. The error is reported for runs with and without data scale augmentation, the "+" denotes the use of data scale augmentation. Learnable scale approaches perform on par with the non-learnable scale baseline SESN [2].

Model	MNIST-Scale	MNIST-Scale+	# Params.
CNN	2.02 ± 0.07	1.60 ± 0.09	495k
SiCNN	2.02 ± 0.14	1.59 ± 0.03	497k
SI-ConvNet	1.82 ± 0.11	1.59 ± 0.10	495k
SEVF	2.12 ± 0.13	1.81 ± 0.09	495k
DSS	1.97 ± 0.08	1.57 ± 0.09	475k
SS-CNN	1.84 ± 0.10	1.76 ± 0.07	494k
SESN (Hermite)	1.68 ± 0.06	1.42 ± 0.07	495k
DISCO	1.52 ± 0.06	1.35 ± 0.05	495k
Ours (Learn ISR)	1.72 ± 0.05	1.44 ± 0.09	495k
Ours (Learn Spacings)	1.70 ± 0.10	1.50 ± 0.08	495k
Ours (Learn Scales Directly)	1.74 ± 0.06	1.50 ± 0.08	495k

Table 5.6: Learned scale parameters of our model on MNIST-Scale compared to values chosen in SESN [2]. The "+" denotes the use of data scale augmentation. The learned scales of all scale learning methods are much wider than the initialized scales in SESN [2] and DISCO [13], especially when scale data augmentation is used.

Model	Scale 1	Scale 2	Scale 3	Scale 4
SESN	1.5	1.89	2.38	3
DISCO	1.8	2.27	2.86	3.6
Ours (Learn ISR)	1.390 ± 0.016	1.890 ± 0.061	2.572 ± 0.164	3.503 ± 0.338
Ours (Learn Spacing)	1.390 ± 0.011	1.930 ± 0.099	2.614 ± 0.300	3.776 ± 0.600
Ours (Learn Scales Directly)	1.375 ± 0.008	1.889 ± 0.054	2.383 ± 0.044	3.297 ± 0.086
Ours (Learn ISR)+	1.381 ± 0.013	2.066 ± 0.012	3.092 ± 0.038	4.627 ± 0.095
Ours (Learn Spacing)+	1.373 ± 0.013	1.859 ± 0.069	2.847 ± 0.091	4.646 ± 0.306
Ours (Learn Scales Directly)+	1.360 ± 0.014	1.741 ± 0.062	2.485 ± 0.050	3.775 ± 0.083

for more advanced data scale distributions it might be essential to find a way to additionally learn or adapt the number of internal scales based on a heuristic.

REFERENCES

[1] Y. Xu, T. Xiao, J. Zhang, K. Yang, and Z. Zhang. "Scale-Invariant Convolutional Neural Networks". In: *CoRR* abs/1411.6369 (2014). arXiv: 1411.6369. URL: http://arxiv.org/abs/1411.6369.

[2] I. Sosnovik, M. Szmaja, and A. W. M. Smeulders. "Scale-Equivariant Steerable Networks". In: *CoRR* abs/1910.11093 (2019). arXiv: 1910.11093. URL: http://arxiv.org/abs/1910.11093.

[3] S. L. Pintea, N. Tomen, S. F. Goes, M. Loog, and J. C. van Gemert. "Resolution learning in deep convolutional networks using scale-space theory". In: *CoRR* abs/2106.03412 (2021). arXiv: 2106.03412. URL: https://arxiv.org/abs/2106.03412.

[4] L. M. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever. "Scale and the differential structure of images". In: *Image and Vision Computing* 10.6 (1992). Information Processing in Medical Imaging, pp. 376–388. ISSN: 0262-8856. DOI: https://doi.org/10.1016/0262-8856(92)90024-W. URL: https://www.sciencedirect.com/science/article/pii/026288569290024W.

[5] T. Lindeberg and J.-O. Eklundh. "Scale-space primal sketch: construction and experiments". In: *Image and Vision Computing* 10.1 (1992), pp. 3–18. ISSN: 0262-8856. DOI: https://doi.org/10.1016/0262-8856(92)90079-I. URL: https://www.sciencedirect.com/science/article/pii/026288569290079I.

[6] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. "Learning Hierarchical Features for Scene Labeling". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2013), pp. 1915–1929. DOI: 10.1109/TPAMI.2012.231.

[7] D. Marcos, B. Kellenberger, S. Lobry, and D. Tuia. "Scale equivariance in CNNs with vector fields". In: *CoRR* abs/1807.11783 (2018). arXiv: 1807.11783. URL: http://arxiv.org/abs/1807.11783.

[8] A. Kanazawa, A. Sharma, and D. W. Jacobs. "Locally Scale-Invariant Convolutional Neural Networks". In: *CoRR* abs/1412.5104 (2014). arXiv: 1412.5104. URL: http://arxiv.org/abs/1412.5104.

[9] R. Ghosh and A. K. Gupta. "Scale Steerable Filters for Locally Scale-Invariant Convolutional Neural Networks". In: *CoRR* abs/1906.03861 (2019). arXiv: 1906.03861. URL: http://arxiv.org/abs/1906.03861.

[10] H. Naderi, L. Goli, and S. Kasaei. "Scale Equivariant CNNs with Scale Steerable Filters". In: *2020 International Conference on Machine Vision and Image Processing (MVIP)*. 2020, pp. 1–5.

[11] W. Zhu, Q. Qiu, A. R. Calderbank, G. Sapiro, and X. Cheng. "Scale-Equivariant Neural Networks with Decomposed Convolutional Filters". In: *CoRR* abs/1909.11193 (2019). arXiv: 1909.11193. URL: http://arxiv.org/abs/1909.11193.

[12] T. Lindeberg. "Scale-covariant and scale-invariant Gaussian derivative networks". In: *CoRR* abs/2011.14759 (2020). arXiv: 2011.14759. URL: https://arxiv.org/abs/2011.14759.

[13] I. Sosnovik, A. Moskalev, and A. W. M. Smeulders. "DISCO: accurate Discrete Scale Convolutions". In: *CoRR* abs/2106.02733 (2021). arXiv: 2106.02733. URL: https://arxiv.org/abs/2106.02733.

[14] N. Saldanha. "Frequency learning for structured CNN filters with Gaussian fractional derivatives". MA thesis. Delft University of Technology, 2021.

[15] N. Tomen, S.-L. Pintea, and J. Van Gemert. "Deep Continuous Networks". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 10324–10335. URL: https://proceedings.mlr.press/v139/tomen21a.html.

[16] Y. Yang, S. Dasmahapatra, and S. Mahmoodi. *Scale-Equivariant UNet for Histopathology Image Segmentation*. 2023. arXiv: 2304.04595 [cs.CV].

[17] G. W. Benton, M. Finzi, P. Izmailov, and A. G. Wilson. "Learning Invariances in Neural Networks". In: *CoRR* abs/2010.11882 (2020). arXiv: 2010.11882. URL: https://arxiv.org/abs/2010.11882.

[18] Z. Sun and T. Blu. *Empowering Networks With Scale and Rotation Equivariance Using A Similarity Convolution*. 2023. arXiv: 2303.00326 [cs.CV].

[19] S. Kullback and R. A. Leibler. "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86. DOI: 10.1214/aoms/1177729694. URL: https://doi.org/10.1214/aoms/1177729694.

[20] Y. Xu, T. Xiao, J. Zhang, K. Yang, and Z. Zhang. "Scale-Invariant Convolutional Neural Networks". In: *CoRR* abs/1411.6369 (2014). arXiv: 1411.6369. URL: http://arxiv.org/abs/1411.6369.

[21] D. Worrall and M. Welling. "Deep Scale-spaces: Equivariance Over Scale". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 7364–7376. URL: http://papers.nips.cc/paper/8956-deep-scale-spaces-equivariance-over-scale.pdf.

APPENDICES

5.A DATASET DESCRIPTION

Dynamic Scale MNIST Dynamic Scale MNIST pads the original 28x28 images from the MNIST dataset [1] to 168x168 pixels and then on initialisation of the dataset, an independent scale for each sample is drawn from the chosen scale distribution. Only scales e larger than 1 are sampled during training time to prevent the influence of information loss which occurs when downsampling the data. Since each digit is upsampled upon accessing no additional storage is needed to use this dataset for various scale distributions. After initialisation the dataset is normalised.

Additionally, this dataset can also be used to evaluate across a range of scales by sampling each test digit individually on multiple scales. The scales to evaluate are rounded to the nearest half-octave of 2. The number to evaluate is determined by the range of octaves times 10. Thus for Fig. 5.4, 45 scales are sampled between $2^{-0.5}$ and $2^{3.5}$ in a logarithmic manner. The underlying MNIST dataset [1] is split into 10k training samples, 5k validation samples, and 50k test samples and 3 different realisations are generated and fixed.

MNIST-Scale The images in the MNIST-Scale dataset are rescaled images of the MNIST dataset [1]. The scales are sampled from a Uniform distribution in the range of 0.3 - 1.0 of the original size and padded back to the original resolution of 28x28 pixels. The dataset is split into 10k training samples, 2k validation samples and 50k test samples and 6 realisations are made.

REFERENCES

[1] L. Deng. "The mnist database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

6

FlexConv: Continuous Kernel Convolutions with Differentiable Kernel Sizes

When designing Convolutional Neural Networks (CNNs), one must select the size of the convolutional kernels before training. Recent works show CNNs benefit from different kernel sizes at different layers, but exploring all possible combinations is unfeasible in practice. A more efficient approach is to learn the kernel size during training. However, existing works that learn the kernel size have a limited bandwidth. These approaches scale kernels by dilation, and thus the detail they can describe is limited. In this work, we propose FlexConv, a novel convolutional operation with which high bandwidth convolutional kernels of learnable kernel size can be learned at a fixed parameter cost. FlexNets model long-term dependencies without the use of pooling, achieve state-of-the-art performance on several sequential datasets, outperform recent works with learned kernel sizes, and are competitive with much deeper ResNets on image benchmark datasets. Additionally, FlexNets can be deployed at higher resolutions than those seen during training. To avoid aliasing, we propose a novel kernel parameterization with which the frequency of the kernels can be analytically controlled. Our novel kernel parameterization shows higher descriptive power and faster convergence speed than existing parameterizations. This leads to important improvements in classification accuracy.

This chapter has been published as:
D.W. Romero, R. Bruintjes, J.M. Tomczak, E.J. Bekkers, M. Hoogendoorn, J.C. van Gemert. "FlexConv: Continuous Kernel Convolutions with Differentiable Kernel Sizes". In: *Proceedings of the 10th International Conference on Learning Representations (ICLR 2022)*, Apr. 2022.

Code available at:
https://github.com/rjbruin/flexconv



6.1 INTRODUCTION

The kernel size of a convolutional layer defines the region from which features are computed, and is a crucial choice in their design. Commonly, small kernels (up to 7px) are used almost exclusively and are combined with pooling to model long term dependencies [1–4]. Recent works indicate, however, that CNNs benefit from using convolutional kernels (*i*) of varying size at different layers [5, 6], and (*ii*) at the same resolution of the data [7–9]. Unfortunately, most CNNs represent convolutional kernels as tensors of discrete weights and their size must be fixed prior to training. This makes exploring different kernel sizes at different layers difficult and time-consuming due to (*i*) the large search space, and (*ii*) the large number of weights required to construct large kernels.

A more efficient way to tune different kernel sizes at different layers is to *learn* them during training. Existing methods define a *discrete* weighted set of basis functions, e.g., shifted Delta-Diracs (Fig. 6.2b, [10]) or Gaussian functions (Fig. 6.2c, [5, 11, 12]). During training they learn dilation factors over the basis functions to increase the kernel size, which crucially limits the bandwidth of the resulting kernels.

In this work, we present the *Flexible Size Continuous Kernel Convolution* (FlexConv), a convolutional layer able to learn *high bandwidth* convolutional kernels of varying size during training (Fig. 6.1). Instead of using discrete weights, we provide a *continuous parameterization* of convolutional kernels via a small neural network [9]. This parameterization allows us to model continuous functions of arbitrary size with a fixed number of parameters. By multiplying the response of the neural network with a Gaussian mask, the size of the kernel can be learned during training (Fig. 6.2a). This allows us to produce detailed kernels of small sizes (Section 6.1), and tune kernel sizes efficiently.

FlexConvs can be deployed at higher resolutions than those observed during training, simply by using a more densely sampled grid of kernel indices. However, the high bandwidth of the kernel can lead FlexConv to learn kernels that show aliasing at higher resolutions, if the kernel bandwidth exceeds the Nyquist frequency. To solve this problem, we propose to parameterize convolutional kernels as *Multiplicative Anisotropic Gabor Networks* (MAGNets). MAGNets are a new class of Multiplicative Filter Networks [13] that allows us to analyze and control the frequency spectrum of the generated kernels. We use this analysis to regularize FlexConv against aliasing. With this regularization, FlexConvs can be directly deployed at higher resolutions with minimal accuracy loss. Furthermore, MAGNets provide higher descriptive power and faster convergence speed than existing continuous kernel parameterizations [9, 14, 15]. This leads to important improvements in classification accuracy (Section 6.4).

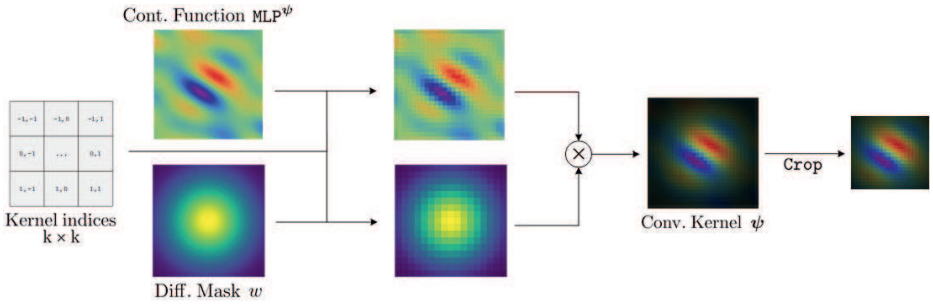
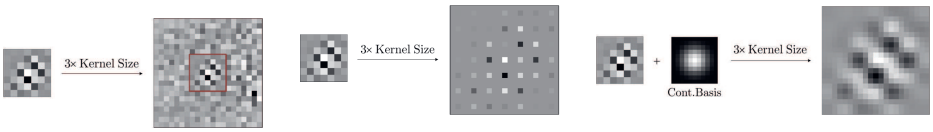


Figure 6.1: The Flexible Size Continuous Kernel Convolution (FlexConv). FlexConv defines convolutional kernels as the multiplication of a continuous convolutional kernel $\texttt{MLP}^{\boldsymbol{\psi}}$, with a Gaussian mask of local support w_{gauss}: $\boldsymbol{\psi}(x, y) = w_{\text{gauss}}(x, y; \boldsymbol{\theta}_{\text{mask}}) \cdot \texttt{MLP}^{\boldsymbol{\psi}}(x, y)$. By learning the parameters of the mask, the size of the convolutional kernel can be optimized during training.

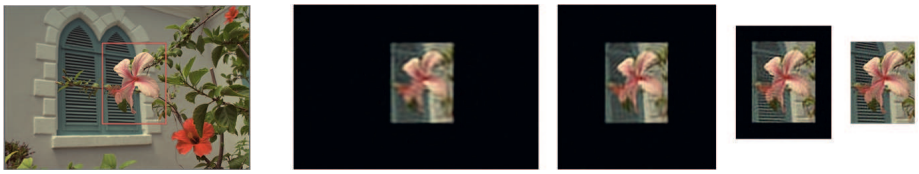


(a) FlexConv kernels (ours)

(b) Dilation / deformation [10]

(c) (Learnable) parametric dilation [5]

Figure 6.2: Existing approaches increase the size of convolutional kernels via (learnable) parametric dilations, e.g., by deformation (b) or by Gaussian blur (c). However, dilation limits the bandwidth of the dilated kernel and with it, the amount of detail it can describe. Contrarily, FlexNets extend their kernels by passing a larger vector of positions to the neural network parameterizing them. As a result, FlexConvs are able to learn *high bandwidth* convolutional kernels of varying size end-to-end (a).



(a) Ground Truth (b) Reconstructions at varying degrees of localization

Figure 6.3: The importance of dynamic sizes in continuous kernel convolutions. Consider a neural network predicting pixel values at each position. If the entire image is considered, the network must use part of its capacity to learn to predict zeros outside of the flower region, which in turn degrades the quality of the approximation in the region of interest (b). Importantly, the better the localization of the flower, the higher the approximation fidelity becomes. FlexNets learn the size of their convolutional kernels at each layer during training, and thus *(i)* use the capacity of the kernel efficiently, *(ii)* converge faster to good approximations, and *(iii)* are faster in execution, via dynamic cropping.

Our experiments show that CNNs with FlexConvs, coined *FlexNets*, achieve state-of-the-art across several sequential datasets, match performance of recent works with learnable kernel sizes with less compute, and are competitive with much deeper ResNets [3] when applied on image benchmark datasets. Thanks to the ability of FlexConvs to generalize across resolutions, FlexNets can be efficiently trained at low-resolution to save compute, e.g., on 16×16 CIFAR images, and be deployed on the original data resolution with marginal accuracy loss, e.g., on 32×32 CIFAR images.

In summary, our **contributions** are:

- We introduce the *Flexible Size Continuous Kernel Convolution* (FlexConv), a convolution operation able to learn *high bandwidth* convolutional kernels of varying size end-to-end.
- Our proposed *Multiplicative Anisotropic Gabor Networks* (MAGNets) allow for analytic control of the properties of the generated kernels. This property allows us to construct analytic alias-free convolutional kernels that generalize to higher resolutions, and to train FlexNets at low resolution and deploy them at higher resolutions. Moreover, MAGNets show higher descriptive power and faster convergence speed than existing kernel parameterizations.
- CNN architectures with FlexConvs (FlexNets) obtain state-of-the-art across

several sequential datasets, and match recent works with learnable kernel size on CIFAR-10 with less compute.

6.2 Related Work

Adaptive kernel sizes. Loog and Lauze [16] regularize the scale of convolutional kernels for filter learning. For image classification, adaptive kernel sizes have been proposed via learnable pixel-wise offsets [10], learnable padding operations [17], learnable dilated Gaussian functions [12, 18–20] and scalable Gaussian derivative filters [5, 6, 21]. These approaches either dilate discrete kernels (Fig. 6.2b), or use discrete weights on dilated basis functions (Fig. 6.2c). Using dilation crucially limits the bandwidth of the resulting kernels. In contrast, FlexConvs are able to construct high bandwidth convolutional kernels of varying size with a fixed parameter count. Larger kernels are obtained simply by passing more positions to the kernel network (Fig. 6.1).

Continuous kernel convolutions. Discrete convolutional kernel parameterizations assign an independent weight to each specific position in the kernel. Continuous convolutional kernels, on the other hand, view convolutional kernels as continuous functions parameterized via a small neural network $\texttt{MLP}^{\boldsymbol{\psi}}: \mathbb{R}^{\mathrm{D}} \rightarrow \mathbb{R}^{\mathrm{N_{out}} \times \mathrm{N_{in}}}$, with D the data dimensionality. This defines a convolutional kernel for which arbitrary input positions can be queried. Continuous kernels have primarily been used to handle irregularly-sampled data *locally*, e.g., molecular data [14, 22] and point-clouds [23–25].

Recently, Romero *et al*. [9] introduced the Continuous Kernel Convolution (CKConv) as a tool to model long-term dependencies. CKConv uses a continuous kernel parameterization to construct convolutional kernels as big as the input signal with a constant parameter cost. Contrarily, FlexConvs jointly learn the convolutional kernel as well as its size. This leads to important advantages in terms of expressivity (Section 6.1), convergence speed and compute costs of the operation.

Implicit neural representations. Parameterizing a convolutional kernel via a neural network can be seen as learning an implicit neural representation of the underlying convolutional kernel [9]. Implicit neural representations construct continuous data representations by encoding data in the weights of a neural network [13, 26, 27].

We replace the SIREN [27] kernel parameterization used by Romero *et al*. [9] by our *Multiplicative Anisotropic Gabor Networks*: a new class of Multiplicative Filter Networks [13]. MFNs allow for analytic control of the resulting representations, and allow us to construct analytic alias-free convolutional kernels. The higher expressivity and convergence speed of MAGNets lead to accuracy improvements in CNNs using them as kernel parameterization.

6.3 METHOD

In this section, we introduce our approach. First, we introduce FlexConv and the Gaussian mask. Next, we introduce our Multiplicative Anisotropic Gabor Networks (MAGNets) and provide a description of our regularization technique used to control the spectral components of the generated kernel.

6.3.1 FLEXIBLE SIZE CONTINUOUS KERNEL CONVOLUTION (FLEXCONV)

To learn the kernel size during training, FlexConvs define their convolutional kernels $\boldsymbol{\psi}$ as the product of the output of a neural network $\texttt{MLP}^{\boldsymbol{\psi}}$ with a Gaussian mask of local support. The neural network $\texttt{MLP}^{\boldsymbol{\psi}}$ parametrizes the kernel, and the Gaussian mask parametrizes its size (Fig. 6.1).

Anisotropic Gaussian mask. Let $G(x; \mu_X, \sigma_X^2) := \exp\{-\frac{1}{2}\sigma_X^{-2}(x-\mu_X)^2\}$ be a Gaussian function parameterized by a mean-variance tuple (μ_X, σ_X^2). The anisotropic Gaussian mask is defined as:

$$w_{\text{gauss}}(x, y; \{\mu_X, \sigma_X^2, \mu_Y, \sigma_Y^2\}) = G(x; \mu_X, \sigma_X^2) G(y; \mu_Y, \sigma_Y^2). \tag{6.1}$$

By learning (μ_X, σ_X^2) and (μ_Y, σ_Y^2) independently, anisotropic non-centered windows can be learned.

6.3.2 MULTIPLICATIVE ANISOTROPIC GABOR NETWORKS (MAGNETS)

In this section, we formalize our proposed parameterization for the kernel $\texttt{MLP}^{\boldsymbol{\psi}}$. We start by introducing Multiplicative Filter Networks [13], and present our MAGNets next.

Multiplicative Filter Networks (MFNs). Recently, Fathony *et al*. [13] proposed to construct implicit neural representations as the linear combination of exponentially many basis functions g:

$$\mathrm{h}^{(1)} = \mathrm{g}\big([x,y];\boldsymbol{\theta}^{(1)}\big) \qquad \mathrm{g}:\mathbb{R}^2 \rightarrow \mathbb{R}^{\mathrm{N_{hid}}} \tag{6.2}$$

$$\mathrm{h}^{(l)} = \big(\mathbf{W}^{(l)}\mathrm{h}^{(l-1)} + \mathrm{b}^{(l)}\big) \cdot \mathrm{g}\big([x,y];\boldsymbol{\theta}^{(l)}\big) \qquad \mathbf{W}^{(l)} \in \mathbb{R}^{\mathrm{N_{hid}}\times\mathrm{N_{hid}}}, \mathrm{b}^{(l)} \in \mathbb{R}^{\mathrm{N_{hid}}} \tag{6.3}$$

$$\boldsymbol{\psi}(x,y) = \mathbf{W}^{(\mathrm{L})}\mathrm{h}^{(\mathrm{L}-1)} + \mathrm{b}^{(\mathrm{L})} \qquad \mathbf{W}^{(\mathrm{L})} \in \mathbb{R}^{\mathrm{N}\times\mathrm{N_{hid}}}, \mathrm{b}^{(\mathrm{L})} \in \mathbb{R}^{\mathrm{N}} \tag{6.4}$$

where $\{\boldsymbol{\theta}^{(l)}, \boldsymbol{W}^{(l)}, \mathrm{b}^{(l)}\}$ depict the learnable parameters of the bases and the affine transformations, and $\mathrm{N}, \mathrm{N_{hid}}$ depict the number of output and hidden channels, respectively. Depending on the selection of g, MFNs obtain approximations comparable to those of SIRENs [27] with faster convergence rate. The most successful instantiation of MNFs are the *Multiplicative Gabor Network* (MGN): MFNs constructed with isotropic Gabor functions as basis g (in Eq. 6.2):

$$\mathrm{g}\big([x,y];\boldsymbol{\theta}^{(l)}\big) = \exp\Big(-\frac{\boldsymbol{\gamma}^{(l)}}{2}\Big[\big(x-\boldsymbol{\mu}^{(l)}\big)^2 + \big(y-\boldsymbol{\mu}^{(l)}\big)^2\Big]\Big)\,\mathrm{Sin}\big(\mathbf{W}_{\mathrm{g}}^{(l)}\cdot[x,y] + \mathrm{b}_{\mathrm{g}}^{(l)}\big), \tag{6.5}$$

$$\boldsymbol{\theta}^{(l)} = \big\{\boldsymbol{\gamma}^{(l)} \in \mathbb{R}^{\mathrm{N_{hid}}}, \boldsymbol{\mu}^{(l)} \in \mathbb{R}^{\mathrm{N_{hid}}}, \mathbf{W}_{\mathrm{g}}^{(l)} \in \mathbb{R}^{\mathrm{N_{hid}}\times 2}, \mathrm{b}_{\mathrm{g}}^{(l)} \in \mathbb{R}^{\mathrm{N_{hid}}}\big\}. \tag{6.6}$$

Note that, by setting $\mathrm{N}=\mathrm{N_{out}}\times\mathrm{N_{in}}$, an MFN can parameterize a convolutional kernel with $\mathrm{N_{in}}$ input and $\mathrm{N_{out}}$ output channels. Fathony *et al*. [13] show that MFNs are equivalent to a linear combination of exponentially many basis functions g. This allows us to analytically derive properties of MFN representations, and plays a crucial role in the derivation of alias-free MAGNets (Section 6.3.3).

Multiplicative Anisotropic Gabor Networks (MAGNets). Our MAGNet formulation is based on the observation that isotropic Gabor functions, i.e., with equal γ for the horizontal and vertical directions, are undesirable as basis for the construction of MFNs. Whenever a frequency is required along a certain direction, an isotropic Gabor function automatically introduces that frequency in both directions. As a result, other bases must counteract this frequency in the direction where the frequency is not required, and thus the capacity of the MFN is not used optimally [28].

Following the original formulation of the 2D Gabor functions [28], we alleviate

this limitation by using anisotropic Gabor functions instead:

$$\mathrm{g}([x,y];\boldsymbol{\theta}^{(l)}) = \exp\Big(-\frac{1}{2}\Big[\Big(\boldsymbol{\gamma}_{\mathrm{X}}^{(l)}\big(x-\boldsymbol{\mu}_{\mathrm{X}}^{(l)}\big)\Big)^2+\Big(\boldsymbol{\gamma}_{\mathrm{Y}}^{(l)}\big(y-\boldsymbol{\mu}_{\mathrm{Y}}^{(l)}\big)\Big)^2\Big]\Big)\,\mathrm{Sin}\big(\mathbf{W}_{\mathrm{g}}^{(l)}[x,y]+\mathrm{b}_{\mathrm{g}}^{(l)}\big) \tag{6.7}$$

$$\boldsymbol{\theta}^{(l)} = \Big\{\boldsymbol{\gamma}_{\mathrm{X}}^{(l)}\in\mathbb{R}^{\mathrm{N_{hid}}},\boldsymbol{\gamma}_{\mathrm{Y}}^{(l)}\in\mathbb{R}^{\mathrm{N_{hid}}},\boldsymbol{\mu}_{\mathrm{X}}^{(l)}\in\mathbb{R}^{\mathrm{N_{hid}}},\boldsymbol{\mu}_{\mathrm{Y}}^{(l)}\in\mathbb{R}^{\mathrm{N_{hid}}},\mathbf{W}_{\mathrm{g}}^{(l)}\in\mathbb{R}^{\mathrm{N_{hid}}\times 2},\mathrm{b}_{\mathrm{g}}^{(l)}\in\mathbb{R}^{\mathrm{N_{hid}}}\Big\}. \tag{6.8}$$

The resulting *Multiplicative Anisotropic Gabor Network* (MAGNet) obtains better control upon frequency components introduced to the approximation, and demonstrates important improvements in terms of descriptive power and convergence speed (Section 6.4).

MAGNet initialization. Fathony *et al*. [13] proposes to initialize MGNs by drawing the size of the Gaussian envelopes, i.e., the $\boldsymbol{\gamma}^{(l)}$ term, from a Gamma$(\alpha\cdot \mathrm{L}^{-1},\beta)$ distribution at every layer $l\in[1,..,\mathrm{L}-1]$. We observe however that this initialization does not provide much variability on the initial extension of the Gaussian envelopes and in fact, most of them cover a large portion of the space at initialization. To stimulate diversity, we initialize the $\{\boldsymbol{\gamma}_{\mathrm{X}}^{(l)},\boldsymbol{\gamma}_{\mathrm{Y}}^{(l)}\}$ terms by a Gamma$(\alpha l^{-1},\beta)$ distribution at the l-th layer. We observe that our proposed initialization consistently leads to better accuracy than the initialization of Fathony *et al*. [13] across all tasks considered. (Section 6.4).

6

6.3.3 Analytic Alias-free MAGNets

FlexConvs can be deployed at higher resolutions than those observed during training, simply by sampling the underlying continuous representation of the kernel more densely, and accounting for the change in sampling rate. Consider a D-dimensional input signal $f_{\mathrm{r}^{(1)}}$ with resolution $\mathrm{r}^{(1)}$. FlexConv learns a kernel $\boldsymbol{\psi}_{\mathrm{r}^{(1)}}$ that can be inferred at a higher resolution $\mathrm{r}^{(2)}$ [9]:

$$\Big(f_{\mathrm{r}^{(2)}} * \boldsymbol{\psi}_{\mathrm{r}^{(2)}}\Big) \approx \left(\frac{\mathrm{r}^{(1)}}{\mathrm{r}^{(2)}}\right)^{\mathrm{D}}\Big(f_{\mathrm{r}^{(1)}} * \boldsymbol{\psi}_{\mathrm{r}^{(1)}}\Big). \tag{6.9}$$

Note however, that Eq. 6.9 holds *approximately*. This is due to aliasing artifacts which can appear if the frequencies in the learned kernel surpass the Nyquist criterion of the target resolution. Consequently, an anti-aliased parameterization is vital to construct kernels that generalize well to high resolutions.

Towards alias-free implicit neural representations. We observe that SIRENs as well as unconstrained MFNs and MAGNets exhibit aliasing when deployed on resolutions higher than the training resolution, which hurts performance of the model. An example kernel with aliasing is shown in Fig. 6..8.

To combat aliasing, we would like to control the representation learned by MAGNets. MAGNets –and MFNs in general– construct implicit neural representations that can be seen as a *linear combination of basis functions*. This property allows us to analytically derive and study the properties of the resulting neural representation. Here, we use this property to derive the maximum frequency of MAGNet-generated kernels, so as to regularize MAGNets against aliasing during training. We analytically derive the maximum frequency of a MAGNet, and penalize it whenever it exceeds the Nyquist frequency of the training resolution. We note that analytic derivations are difficult for other implicit neural representations, e.g., SIRENs, due to stacked layer-wise nonlinearities.

Maximum frequency of MAGNets. The maximum frequency component of a MAGNet is given by:

$$f^{+}_{\text{MAGNet}} = \sum_{l=1}^{\text{L}} \max_{i_l} \left(\left(\max_{j} \frac{\mathbf{W}^{(l)}_{\text{g},i_l,j}}{2\pi} \right) + \frac{\sigma_{\text{cut}} \min\{\boldsymbol{\gamma}^{(l)}_{\text{X},i_l}, \boldsymbol{\gamma}^{(l)}_{\text{Y},i_l}\}}{2\pi} \right), \tag{6.21}$$

where L corresponds to the number of layers, $\mathbf{W}^{(l)}_{\text{g}}, \boldsymbol{\gamma}^{(l)}_{\text{X}}, \boldsymbol{\gamma}^{(l)}_{\text{Y}}$ to the MAGNet parameters as defined in Eq. 6.8, and σ_{cut}=2 $\cdot$ `stdev` to the cut-off frequency of the Gaussian envelopes in the Gabor filters. A formal treatment as well as the derivations can be found in Section 6.A.1.

Effect of the FlexConv mask. The Gaussian mask used to localize the response of the MAGNet also has an effect on the frequency spectrum. Hence, the maximum frequency of a FlexConv kernel is:

$$f^{+}_{\text{FlexConv}} = f^{+}_{\text{MAGNet}} + f^{+}_{w_{\text{gauss}}}, \text{ with } f^{+}_{w_{\text{gauss}}} = \frac{\sigma_{\text{cut}}}{\max\{\sigma_{\text{X}}, \sigma_{\text{Y}}\} 2\pi}. \tag{6.22}$$

Here, $\sigma_{\text{X}}, \sigma_{\text{Y}}$ correspond to the mask parameters (Eq. 6.1). Intuitively, multiplication with the mask blurs in the frequency domain, as it is equivalent to convolution with the Fourier transform of the mask.

Aliasing regularization of FlexConv kernels. With the analytic derivation of f^{+}_{FlexConv} we penalize the generated kernels to have frequencies smaller or equal

to their Nyquist frequency $f_{\text{Nyq}}(k)$ via:

$$\mathscr{L}_{\text{HF}} = ||\max\{f^{+}_{\text{FlexConv}}, f_{\text{Nyq}}(k)\} - f_{\text{Nyq}}(k)||^2, \quad \text{with} \quad f_{\text{Nyq}}(k) = \tfrac{k-1}{4}. \tag{6.25}$$

Here, k depicts the size of the FlexConv kernel before applying the Gaussian mask, and is equal to the size of the input signal. In practice, we implement Eq. 6.25 by regularizing the individual MAGNet layers, as is detailed in Section 6.A.2. To verify our method, Fig. 6..8 (Section 6.A.1) shows that the frequency components of FlexNet kernels are properly regularized for aliasing.

6.4 Experiments

We evaluate FlexConv across classification tasks on sequential and image benchmark datasets, and validate the ability of MAGNets to approximate complex functions. A complete description of the datasets used is given in Section 6.B. Section 6.D.2 reports the parameters used in all our experiments.[1]

6.4.1 What kind of functions can MAGNets approximate?

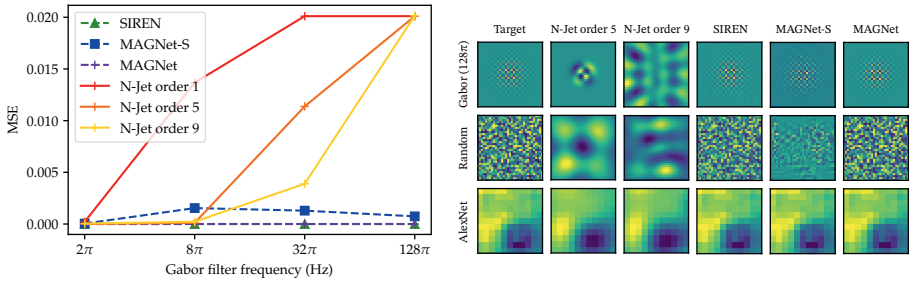


Figure 6.4: *Left*: Final MSE after fitting each model to Gabor filters of different frequencies. N-Jets cannot fit high frequencies. *Right*: Kernels learned by each model. SIREN and MAGNet can fit all targets. MAGNet-S: a small MAGNet of size akin to N-Jets, still does well on the Gabor and AlexNet targets.

[1] Our code is publicly available at https://github.com/rjbruin/flexconv.

Bandwidth of methods with learnable sizes. First, we compare the bandwidth of MAGNet against N-Jet [5] by optimizing each to fit simple targets: *(i)* Gabor filters of known frequency, *(ii)* random noise and *(iii)* an a 11×11 AlexNet kernel from the first layer [29]. Fig. 6.4 shows that, even with 9 orders of Gaussian derivatives, N-Jets cannot fit high frequency signals in large kernels. Crucially, N-Jet models require many Gaussian derivative orders to model high frequency signals in large kernels: a hyperparameter which proportionally increases their inference time and parameter count. MAGNets, on the other hand, accurately model large high frequency signals. This allows FlexNets to learn large kernels with high frequency components.

Expressivity of `MLP` parameterizations. Next, we compare the descriptive power and convergence speed of MAGNets, Gabor MFNs, Fourier MFNs and SIRENs for image approximation. To this end, we fit the images in the Kodak dataset [30] with each of these methods. Our results (Table 6.C.1) show that MAGNets outperform all other methods, and converge faster to good approximations.

6.4.2 Classification Tasks

Network specifications. Here, we specify our networks for all our classification experiments. We parameterize all our convolutional kernels as the superposition of a 3-layer MAGNet and a learnable anisotropic Gaussian mask. We construct two network instances for sequential and image datasets respectively: FlexTCNs and FlexNets. Both are constructed by taking the structure of a baseline network –TCN [33] or CIFARResNet [3]–, removing all internal pooling layers, and replacing convolutional kernels by FlexConvs. The FlexNet architecture is shown in Fig. 6.D.1 and varies only in the number of channels and blocks, e.g., FlexNet-16 has 7 blocks. Akin to Romero *et al*. [9] we utilize the Fourier theorem to speed up convolutions with large kernels.

Mask initialization. We initialize the FlexConv masks to be small. Preliminary experiments show this leads to better performance, faster execution, and faster training convergence. For sequences, the mask center is initialized at the last kernel position to prioritize the last information seen.

Time series and sequential data. First we evaluate FlexTCNs on sequential classification datasets, for which long-term dependencies play an important

Table 6.1: Test accuracy and ablation studies on sMNIST, pMNIST, sCIFAR10 and npCIFAR10.

MODEL	SIZE	SMNIST	PMNIST	SCIFAR10	NPCIFAR10
DilRNN [31]	44K	98.0	96.1	-	-
IndRNN [32]	83K	99.0	96.0	-	-
TCN [33]	70K	99.0	97.2	-	-
r-LSTM [34]	0.5M	98.4	95.2	72.2	-
Self-Att. [34]	0.5M	98.9	97.9	62.2	-
TrellisNet [35]	8M	99.20	98.13	73.42	-
URLSTM [36]	-	99.28	96.96	71.00	-
URGRU + Zoneout [36]	-	99.27	96.51	**74.40**	-
HiPPO [37]	0.5M	-	**98.30**	-	-
Lipschitz RNN [38]	158K	99.4	97.3	64.2	59.0
coRNN [39]	134K	**99.4**	97.3	-	59.0
UnICORNN [40]	135K	-	98.4	-	**62.4**
pLMU [41]	165K	-	98.49	-	-
CKCNN-2	98K	99.31	98.00	62.25	60.5
CKCNN-2-Big	1M	99.32	98.54	63.74	62.2
$\text{CKTCN}_{\text{FOURIER}}$-2	105K	99.44	98.40	68.28	66.26
$\text{CKTCN}_{\text{GABOR}}$-2	106K	99.52	98.38	69.26	67.37
$\text{CKTCN}_{\text{MAGNET}}$-2	105K	**99.55**	**98.57**	**74.58**	**67.52**
FlexTCN-2	108K	**99.60**	**98.61**	**78.99**	**67.11**
FlexTCN-4	241K	**99.60**	**98.72**	**80.26**	**67.42**
FlexTCN-6	375K	**99.62**	**98.63**	**80.82**	**69.87**
$\text{FlexTCN}_{\text{SIREN}}$-6	343K	99.03	95.36	69.24	57.27
$\text{FlexTCN}_{\text{Fourier}}$-6	370K	99.49	97.97	74.79	67.35
$\text{FlexTCN}_{\text{Gabor}}$-6	373K	99.50	98.37	78.36	67.56
$\text{FlexTCN}_{\text{MAGNet}}$-6	375K	**99.62**	**98.63**	**80.82**	**69.87**

role. We validate our approach on intrinsic discrete data: *sequential MNIST*, *permuted MNIST* [42], *sequential CIFAR10* [31], *noise-padded CIFAR10* [43], as well as time-series data: *CharacterTrajectories* (CT) [44], *SpeechCommands* [45] with raw waveform (SC_raw) and MFCC input representations (SC).

Our results are summarized in Tables 6.1, 6.2. FlexTCNs with two residual blocks obtain state-of-the-art results on all tasks considered. In addition, depth further improves performance. FlexTCN-6 improves the current state-of-the-art on sCIFAR10 and npCIFAR10 by more than 6%. On the difficult SC_raw dataset –with sequences of length 16000–, FlexTCN-6 outperform the previous state-of-the-art by 20.07%: a remarkable improvement.

Furthermore, we conduct ablation studies by changing the parameterization of $\texttt{MLP}^{\Psi}$, and switching off the learnable kernel size ("CKTCNs") and considering global kernel sizes instead. CKTCNs and FlexTCNs with MAGNet kernels out-

Table 6.2: Test accuracy on CT, SC and SC_raw

Model	Size	CT	SC	SC_raw
GRU-ODE	89k	96.2	44.8	~10.0
GRU-Δt	89k	97.8	20.0	~10.0
GRU-D	89k	95.9	23.9	~10.0
ODE-RNN	89k	97.1	93.2	~10.0
NCDE	89k	98.8	88.5	~10.0
CKCNN	100k	**99.53**	95.27	71.66
$\text{CKTCN}_{\text{Fourier}}$		-	95.65	74.90
$\text{CKTCN}_{\text{Gabor}}$		-	96.66	78.10
$\text{CKTCN}_{\text{MAGNet}}$	105k	**99.53**	**97.01**	**80.69**
FlexTCN-2	105sk	**99.53**	**97.10**	**88.03**
FlexTCN-4	239k	**99.53**	**97.73**	**90.45**
FlexTCN-6	373k	**99.53**	**97.67**	**91.73**
$\text{FlexTCN}_{\text{SIREN}}$-6	370k	-	95.83	85.73
$\text{FlexTCN}_{\text{Fourier}}$-6	342k	-	97.62	91.02
$\text{FlexTCN}_{\text{Gabor}}$-6	373k	-	97.35	91.50
$\text{FlexTCN}_{\text{MAGNet}}$-6	373k	-	**97.67**	**91.73**

Table 6.3: Results on CIFAR-10. Results from *original works and † single run.

Model	Size	CIFAR-10 Acc.	Time (sec/epoch)
CIFARResNet-44	0.66m	92.9*†	22
DCN-σ^{ji}	0.47m	89.7 ± 0.3*	-
N-Jet-CIFARResNet32	0.52m	92.3 ± 0.3*	-
N-Jet-ALLCNN	1.07m	92.5 ± 0.1*	-
FlexNet-16 w/ conv. ($k = 3$)	0.17m	89.5 ± 0.3	41
FlexNet-16 w/ conv. ($k = 33$)	20.0m	78.0 ± 0.3	242
FlexNet-16 w/ N-Jet	0.70m	91.7 ± 0.1	409
CKCNN-16	0.63m	72.1 ± 0.2	68
$\text{CKCNN}_{\text{MAGNet}}$-16	0.67m	86.8 ± 0.6	102
$\text{FlexNet}_{\text{SIREN}}$-16	0.63m	89.0 ± 0.3	89
$\text{FlexNet}_{\text{Gabor}}$-16	0.67m	91.9 ± 0.2	161
$\text{FlexNet}_{\text{Gabor}}$-16 + anis. Gauss.	0.67m	92.0 ± 0.1	147
$\text{FlexNet}_{\text{Gabor}}$-16 + Gabor init.	0.67m	92.0 ± 0.2	150
FlexNet-16	0.67m	92.2 ± 0.1	127

perform corresponding models with all other kernel parameterizations: SIRENs [27], MGNs and MFNs [13]. Moreover, we see a consistent improvement with respect to CKCNNs [9] by using learnable kernel sizes. This shows that both MAGNets and learnable kernel sizes contribute to the performance of FlexTCNs. Note that in 1D, MAGNets are equivalent to MGNs. However, MAGNets consistently perform better than MGNs. This improvement in accuracy is a result of our MAGNet initialization.

Image classification. Next, we evaluate FlexNets for image classification on CIFAR-10 [46]. Additional experiments on Imagenet-32, MNIST and STL-10 can be found in Section 6.C.

Table 6.3 shows our results on CIFAR-10. FlexNets are competitive with pooling-based methods such as CIFARResNet [3] and outperform learnable kernel size method DCNs [6]. In addition, we compare using N-Jet layers of order three (as in [5]) in FlexNets against using MAGNet kernels. We observe that N-Jet layers lead to worse performance, and are significantly slower than FlexConv layers with MAGNet kernels. The low accuracy of N-Jet layers is likely to be linked to the fact that FlexNets do not use pooling. Consequently, N-Jets are forced to learn large kernels with high-frequencies, which we show N-Jets struggle learning in Section 6.4.1.

To illustrate the effect of learning kernel sizes, we also compare FlexNets against FlexNets with large and small discrete convolutional kernels (Table 6.3). Using small kernel sizes is parameter efficient, but is not competitive with FlexNets. Large discrete kernels on the other hand require a copious amount of parameters and lead to significantly worse performance. These results indicate that the best solution is somewhere in the middle and varying kernel sizes can learn the optimal kernel size for the task at hand.

Similar to the sequential case, we conduct ablation studies on image data with learnable, non-learnable kernel sizes and different kernel parameterizations. Table 6.3 shows that FlexNets outperform CKCNNs with corresponding kernel parameterizations. In addition, a clear difference in performance is apparent for MAGNets with respect to other parameterizations. These results corroborate that both MAGNets and FlexConvs contribute to the performance of FlexNets. Moreover, Table 6.3 illustrates the effect of the two contributions of MAGNet over MGN: anisotropic Gabor filters, and our improved initialization. Our results in image data are in unison with our previous results for sequential data (Tables 6.1, 6.2) and illustrate the value of the proposed improvements in MAGNets.

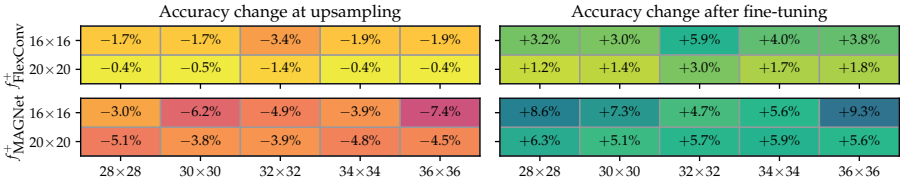


Figure 6.5: Alias-free FlexNet-16 on CIFAR-10. We report change in accuracy between source and target resolutions, directly after upsampling (left) and after fine-tuning (right) (means over five runs).

Table 6.4: Alias-free FlexNets on CIFAR-10.

Model	Size	CIFAR-10 Acc. 16 px	CIFAR-10 Acc. $\Delta_{16\text{px}}$ 32 px
CIFARResNet-44	0.66M	85.8 ± 0.2	-31.6 ± 1.3
FlexNet-16 w/ conv. ($k = 3$)	0.17M	85.3 ± 0.2	-21.2 ± 1.0
FlexNet-16 w/ conv. ($k = 33$)	20.0M	67.7 ± 0.6	-57.1 ± 1.6
FlexNet-16 w/ N-Jets	0.70M	**86.4** ± 0.2	-5.5 ± 1.3
CKCNN-16$_{\text{SIREN}}$	0.63M	45.9 ± 1.0	-15.8 ± 1.2
FlexNet-16$_{\text{SIREN}}$	0.63M	70.4 ± 0.8	-50.0 ± 16.9
FlexNet-16 w/o reg.	0.67M	**86.4** ± 0.4	-34.4 ± 14.3
FlexNet-16 w/ reg. f^+_{MAGNet}	0.67M	**86.5** ± 0.1	-3.8 ± 2.0
FlexNet-16 w/ reg. f^+_{FlexConv}	0.67M	85.1 ± 0.3	**-3.3** ± 0.3

6.4.3 Alias-free FlexNets

Regularizing the FlexConv mask. Though including $f^+_{w_{\text{gauss}}}$ in the frequency analysis of MAGNets is crucial for the accuracy of the derivation, including the FlexConv mask in aliasing regularization is undesirable, as it steers the model to learn large kernels in order to minimize the loss (see Eq. 6.25). However, excluding the mask from regularization could compromise the ability of FlexNet to generalize to higher resolutions. Here, we experiment with this trade-off.

Fig. 6.5 shows accuracy change between ten source and target resolution combinations on CIFAR-10, both for including and excluding the FlexConv mask in the aliasing regularization. We train at the source resolution for 100 epochs,

before testing the model at the target resolution with the upsampling described in Section 6.3.3. Next, we adjust $f_{\text{Nyq}}(k)$ to the target resolution, and finetune each model for 100 epochs at the target resolution.

We find that regularizing just f^{+}_{MAGNet} yields a trade-off. It increases the accuracy difference between low and high resolution inference, but also increases the fine-tune accuracy at the target resolution.We therefore choose to, by default, regularize f^{+}_{MAGNet} only.

Results of our alias-free FlexNet training on CIFAR-10 are in Table 6.4. We observe that the performance of a FlexNet trained without aliasing regularization largely breaks down when the dataset is upscaled. However, with our aliasing regularization most of the performance is retained.

Comparatively, FlexNet retains more of the source resolution performance than FlexNets with N-Jet layers, while baselines degrade drastically at the target resolution. Fig. 6..8 shows the effect of aliasing regularization on the frequency components of FlexConv.

Training at lower resolutions saves compute. We can train alias-free FlexNets at lower resolutions. To verify that this saves compute, we time the first 32 batches of training a FlexNet-7 on CIFAR-10. We compare against training on 16×16 images (downsampled before training). On 16x16 images, each batch takes 179ms (± 7ms). On 32x32 images, each batch takes 222ms (± 9ms). Therefore, we save 24% training time when training FlexNets alias-free at half the native CIFAR-10 resolution.

6.5 Discussion

Learned kernel sizes match conventional priors. Commonly, CNNs use architectures of small kernels and pooling layers. This allows convolutions to build a progressively growing receptive field. With learnable kernel sizes, FlexNet could learn a different prior over receptive fields, e.g., large kernels first, and small kernels next. However, FlexNets learn to increase kernel sizes progressively (Fig. 6.6), and match the network design that has been popular since AlexNet [29].

Mask initialization as a prior for feature importance. The initial values of the FlexConv mask can be used to prioritize information at particular input regions. For instance, initializing the center of mask on the first element of sequential FlexConvs can be used to prioritize information from the far past. This prior is advantageous for tasks such as npCIFAR10. We observe that using this prior on

npCIFAR10 leads to much faster convergence and better results (68.33% acc. w/ FlexTCN-2).

MAGNet regularization as prior induction. MAGNets allow for analytic control of the properties of the resulting representations. We use this property to generate alias-free kernels. However, other desiderata could be induced, e.g., smoothness, for the construction of implicit neural representations.

Benefits of cropping and the influence of `PyTorch`. Dynamic cropping adjust the computational cost of the convolutions on the fly. For a signal of size M and a cropped kernel size k, this incurs in savings from $O(M^{2^D})$ to $O(M^D k^D)$ relative to using global kernel sizes ($O(M^4)$ to $O(M^2k^2)$ in 2D). We test this theoretical speed up in a controlled environment for the Speech Commands and CIFAR-10 datasets. Cropping reduces the per-epoch run time by a factor of 11.8x and 5.5x for Speech Commands and CIFAR-10, respectively. Interestingly, however, both run times become similar if the flag `torch.backends.cudnn.benchmark` is activated, with global kernel sizes being sometimes faster. This is because this flag tells `PyTorch` to optimize the convolution algorithms used under the hood, and some of these CUDA algorithms seem to be faster than our masking strategy on `Python`.

6.6 LIMITATIONS

Dynamic kernel sizes: computation and memory cost of convolutions with large kernels. Performing convolutions with large convolutional kernels is a compute-intensive operation. FlexConvs are initialized with small kernel sizes and their inference cost is relatively small at the start of training. However, despite the cropping operations used to improve computational efficiency (Fig. 6.1, Section 6.1, Table 6.3), the inference time may increase to up to double as the learned masks increase in size. At the cost of more memory, convolutions can be sped up by performing them in the frequency domain. However, we observe that this does not bring gains for the image data considered because FFT convolutions are faster only for very large convolutional kernels (in the order of hundreds of pixels).

Remaining accuracy drop in alias-free FlexNets. Some drop in accuracy is still observed when using alias-free FlexNets at a higher test resolutions (Table 6.4). Although more evidence is needed, this may be caused by aliasing effects introduced by ReLU [47], or changes in the activation statistics of the feature maps

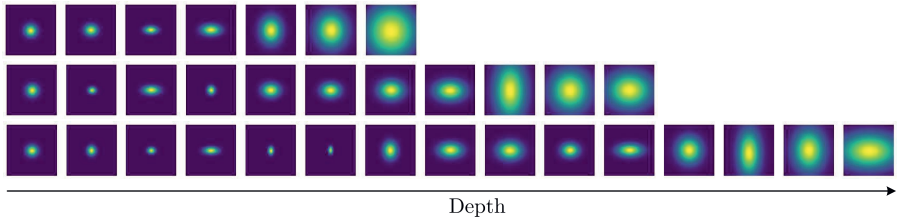


Figure 6.6: Learned FlexConv masks for FlexNets with 3, 5 and 7 residual blocks. FlexNets learn very small kernels at shallow layers, which become larger as a function of depth.

passed to global average pooling [48].

6.7 Conclusion

We propose FlexConv, a convolutional operation able to learn high bandwidth convolutional kernels of varying size during training at a fixed parameter cost. We demonstrate that FlexConvs are able to model long-term dependencies without the need of pooling, and shallow pooling-free FlexNets achieve state-of-the-art performance on several sequential datasets, match performance of recent works with learned kernel sizes with less compute, and are competitive with much deeper ResNets on image benchmark datasets. In addition, we show that our alias-free convolutional kernels allow FlexNets to be deployed at higher resolutions than seen during training with minimal precision loss.

Future work. MAGNets give control over the bandwidth of the kernel. We anticipate that this control has more uses, such as fighting sub-sampling aliasing [49–51]. With the ability to upscale FlexNets to different input image sizes comes the possibility of transfer learning representations between previously incompatible datasets, such as CIFAR-10 and Imagenet. In a similar vein, the automatic adaptation of FlexConv to the kernel sizes required for the task at hand may make it possible to generalize the FlexNet architecture across different tasks and datasets. Neural Architecture Search [52] could see benefits from narrowing the search space to exclude kernel size and pooling layers. In addition, we envisage additional improvements from structural developments of FlexConvs such as attentive FlexNets.

REFERENCES

[1] K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[3] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[4] M. Tan and Q. Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6105–6114.

[5] S. L. Pintea, N. Tomen, S. F. Goes, M. Loog, and J. C. van Gemert. "Resolution learning in deep convolutional networks using scale-space theory". In: *arXiv preprint arXiv:2106.03412* (2021).

[6] N. Tomen, S.-L. Pintea, and J. Van Gemert. "Deep Continuous Networks". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 10324–10335. URL: https://proceedings.mlr.press/v139/tomen21a.html.

[7] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun. "Large kernel matters–improve semantic segmentation by global convolutional network". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4353–4361.

[8] J.-B. Cordonnier, A. Loukas, and M. Jaggi. "On the relationship between self-attention and convolutional layers". In: *arXiv preprint arXiv:1911.03584* (2019).

[9] D. W. Romero, A. Kuzina, E. J. Bekkers, J. M. Tomczak, and M. Hoogendoorn. "CKConv: Continuous Kernel Convolution For Sequential Data". In: *arXiv preprint arXiv:2102.02611* (2021).

[10] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. "Deformable convolutional networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 764–773.

[11] J.-H. Jacobsen, J. Van Gemert, Z. Lou, and A. W. Smeulders. "Structured receptive fields in cnns". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2610–2619.

[12] E. Shelhamer, D. Wang, and T. Darrell. "Blurring the Line Between Structure and Learning to Optimize and Adapt Receptive Fields". In: *ArXiv* abs/1904.11487 (2019).

[13] R. Fathony, A. K. Sahu, D. Willmott, and J. Z. Kolter. "Multiplicative Filter Networks". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=OmtmcPkkhT.

[14] K. Schütt, P.-J. Kindermans, H. E. S. Felix, S. Chmiela, A. Tkatchenko, and K.-R. Müller. "Schnet: A continuous-filter convolutional neural network for modeling quantum interactions". In: *Advances in neural information processing systems*. 2017, pp. 991–1001.

[15] M. Finzi, S. Stanton, P. Izmailov, and A. G. Wilson. "Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data". In: *arXiv preprint arXiv:2002.12880* (2020).

[16] M. Loog and F. Lauze. "Supervised Scale-Regularized Linear Convolutionary Filters". In: *Proceedings of the British Machine Vision Conference (BMVC)*. Ed. by G. B. Tae-Kyun Kim Stefanos Zafeiriou and K. Mikolajczyk. BMVA Press, Sept. 2017, pp. 162.1–162.11. ISBN: 1-901725-60-X. DOI: 10.5244/C.31.162. URL: https://dx.doi.org/10.5244/C.31.162.

[17] S. Han, Z. Meng, Z. Li, J. O'Reilly, J. Cai, X. Wang, and Y. Tong. "Optimizing Filter Size in Convolutional Neural Networks for Facial Action Unit Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[18] Z. Xiong, Y. Yuan, N. Guo, and Q. Wang. "Variational Context-Deformable ConvNets for Indoor Scene Parsing". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[19] D. Tabernik, M. Kristan, and A. Leonardis. "Spatially-Adaptive Filter Units for Compact and Efficient Deep Neural Networks". In: *International Journal of Computer Vision* 128 (Sept. 2020). DOI: 10.1007/s11263-019-01282-1.

[20] D. Nguyen. "Robust deep learning for computer vision to counteract data scarcity and label noise". PhD thesis. Jan. 2020.

[21] T. Lindeberg. "Scale-covariant and scale-invariant Gaussian derivative networks". In: *Scale Space and Variational Methods in Computer Vision :* vol. 12679. Springer Lecture Notes in Computer Science 12679. Not duplicate with DiVA 1505585QC 20210317. Springer Nature, 2021, pp. 3–14. ISBN: 978-3-030-75549-2. DOI: 10.1007/978-3-030-75549-2_1. URL: https://ssvm2021.sciencesconf.org.

[22] 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*. 2017, pp. 3693–3702.

[23] N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley. "Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds". In: *arXiv preprint arXiv:1802.08219* (2018).

[24] S. Wang, S. Suo, W.-C. Ma, A. Pokrovsky, and R. Urtasun. "Deep parametric continuous convolutional neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2589–2597.

[25] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li. "From Points to Parts: 3D Object Detection from Point Cloud with Part-aware and Part-aggregation Network". In: *arXiv preprint arXiv:1907.03670* (2019).

[26] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. "Deepsdf: Learning continuous signed distance functions for shape representation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 165–174.

[27] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. "Implicit neural representations with periodic activation functions". In: *Advances in Neural Information Processing Systems* 33 (2020).

[28] J. Daugman. "Complete discrete 2-D Gabor transforms by neural networks for image analysis and compression". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 36.7 (1988), pp. 1169–1179. DOI: 10.1109/29.1644.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[30] Kodak. *Kodak Dataset*. 1991. URL: http://r0k.us/graphics/kodak/.

[31] S. Chang, Y. Zhang, W. Han, M. Yu, X. Guo, W. Tan, X. Cui, M. Witbrock, M. A. Hasegawa-Johnson, and T. S. Huang. "Dilated recurrent neural networks". In: *Advances in neural information processing systems*. 2017, pp. 77–87.

[32] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao. "Independently recurrent neural network (indrnn): Building a longer and deeper rnn". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 5457–5466.

[33] S. Bai, J. Z. Kolter, and V. Koltun. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling". In: *arXiv preprint arXiv:1803.01271* (2018).

[34] T. H. Trinh, A. M. Dai, M.-T. Luong, and Q. V. Le. "Learning longer-term dependencies in rnns with auxiliary losses". In: *arXiv preprint arXiv:1803.00144* (2018).

[35] S. Bai, J. Z. Kolter, and V. Koltun. "Trellis networks for sequence modeling". In: *arXiv preprint arXiv:1810.06682* (2018).

[36] A. Gu, C. Gulcehre, T. Paine, M. Hoffman, and R. Pascanu. "Improving the gating mechanism of recurrent neural networks". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 3800–3809.

[37] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré. "HiPPO: Recurrent Memory with Optimal Polynomial Projections". In: *arXiv preprint arXiv:2008.07669* (2020).

[38] N. B. Erichson, O. Azencot, A. Queiruga, L. Hodgkinson, and M. W. Mahoney. "Lipschitz recurrent neural networks". In: *arXiv preprint arXiv:2006.12070* (2020).

[39] T. K. Rusch and S. Mishra. "Coupled Oscillatory Recurrent Neural Network (coRNN): An accurate and (gradient) stable architecture for learning long time dependencies". In: *arXiv preprint arXiv:2010.00951* (2020).

[40] T. K. Rusch and S. Mishra. "UnICORNN: A recurrent model for learning very long time dependencies". In: *arXiv preprint arXiv:2103.05487* (2021).

[41] N. Chilkuri and C. Eliasmith. "Parallelizing Legendre Memory Unit Training". In: *arXiv preprint arXiv:2102.11417* (2021).

[42] Q. V. Le, N. Jaitly, and G. E. Hinton. "A simple way to initialize recurrent networks of rectified linear units". In: *arXiv preprint arXiv:1504.00941* (2015).

[43] B. Chang, M. Chen, E. Haber, and E. H. Chi. "AntisymmetricRNN: A dynamical system view on recurrent neural networks". In: *arXiv preprint arXiv:1902.09689* (2019).

[44] A. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh. "The UEA multivariate time series classification archive, 2018". In: *arXiv preprint arXiv:1811.00075* (2018).

[45] P. Warden. "Speech commands: A dataset for limited-vocabulary speech recognition". In: *arXiv preprint arXiv:1804.03209* (2018).

[46] A. Krizhevsky, G. Hinton, *et al. Learning multiple layers of features from tiny images*. Tech. rep. 2009.

[47] C. Vasconcelos, H. Larochelle, V. Dumoulin, R. Romijnders, N. L. Roux, and R. Goroshin. "Impact of Aliasing on Generalization in Deep Convolutional Networks". In: *arXiv preprint arXiv:2108.03489* (2021).

[48] H. Touvron, A. Vedaldi, M. Douze, and H. Jégou. "Fixing the train-test resolution discrepancy". In: *NeurIPS*. 2019.

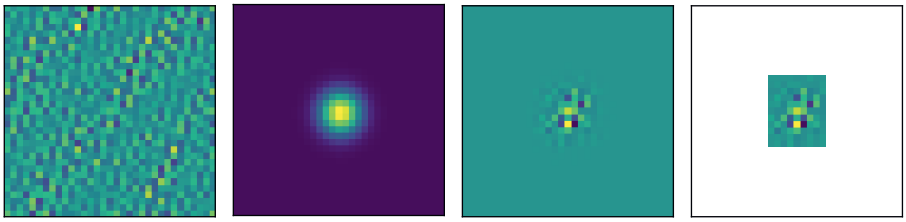
[49] R. Zhang. "Making convolutional networks shift-invariant again". In: *International conference on machine learning*. PMLR. 2019, pp. 7324–7334.

[50] O. S. Kayhan and J. C. v. Gemert. "On Translation Invariance in CNNs: Convolutional Layers Can Exploit Absolute Spatial Location". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[51] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila. "Alias-Free Generative Adversarial Networks". In: *arXiv preprint arXiv:2106.12423* (2021).

[52] B. Zoph and Q. V. Le. "Neural architecture search with reinforcement learning". In: *arXiv preprint arXiv:1611.01578* (2016).

APPENDICES



(a) $\texttt{MLP}^{\boldsymbol{\psi}}$ output (b) $\mathbf{g}([x, y]; \boldsymbol{\theta}^{(l)})$ (c) $\boldsymbol{\psi}(x, y) = \text{(a)} \cdot \text{(b)}$ (d) After cropping

Figure 6..7: Example kernels, generated step by step. FlexConv samples a kernel from $\texttt{MLP}^{\boldsymbol{\psi}}$ (a), which is attenuated by an anistropic Gaussian envelope with learned parameters $\boldsymbol{\theta}^{(l)}$ (b), creating (c) which is cropped to contain only values of > 0.1 (d).

6.A ALIAS-FREE FLEXCONV REGULARIZATION

In this section we provide the complete derivation and analysis for our FlexConv regularization against aliasing. First, we derive the analytic maximum frequency component of a FlexConv kernel. Next, we compute the Nyquist frequency of a FlexConv kernel, and subsequently show how to combine the previous results into a regularization term to train alias-free FlexConvs.

6.A.1 ANALYZING THE FREQUENCY SPECTRUM OF FLEXCONV

In order to make FlexConv alias-free (Section 6.3.3), we need to compute the maximum frequency component of the kernels generated by a MAGNet, so that we can regularize it during training. In this section we analytically derive this maximum frequency component from the parameters of the MAGNet.

Recall that MAGNets generate a kernel $\boldsymbol{\psi}(x, y)$ through of a succession of anisotropic Gabor filters and linear layers (Section 6.3.2, Eqs. 6.2 to 6.7):

$$\begin{aligned}
&\mathrm{h}^{(1)} = \mathrm{g}\big([x, y]; \boldsymbol{\theta}^{(1)}\big) && \mathrm{g} : \mathbb{R}^2 \rightarrow \mathbb{R}^{\mathrm{N_{hid}}} \\
&\mathrm{h}^{(l)} = \big(\mathbf{W}^{(l)}\mathrm{h}^{(l-1)} + \mathrm{b}^{(l)}\big) \cdot \mathrm{g}\big([x, y]; \boldsymbol{\theta}^{(l)}\big) && \mathbf{W}^{(l)} \in \mathbb{R}^{\mathrm{N_{hid}} \times \mathrm{N_{hid}}}, \mathrm{b}^{(l)} \in \mathbb{R}^{\mathrm{N_{hid}}} \\
&\boldsymbol{\psi}(x, y) = \mathbf{W}^{(\mathrm{L})}\mathrm{h}^{(\mathrm{L}-1)} + \mathrm{b}^{(\mathrm{L})} && \mathbf{W}^{(\mathrm{L})} \in \mathbb{R}^{(\mathrm{N_{out}} \times \mathrm{N_{in}}) \times \mathrm{N_{hid}}}, \mathrm{b}^{(\mathrm{L})} \in \mathbb{R}^{(\mathrm{N_{out}} \times \mathrm{N_{in}})}
\end{aligned}$$

$$\mathrm{g}\big([x, y]; \boldsymbol{\theta}^{(l)}\big) = \exp\Big(-\frac{1}{2}\Big[\Big(\boldsymbol{\gamma}_{\mathrm{X}}^{(l)}\big(x - \boldsymbol{\mu}_{\mathrm{X}}^{(l)}\big)\Big)^2 + \Big(\boldsymbol{\gamma}_{\mathrm{Y}}^{(l)}\big(y - \boldsymbol{\mu}_{\mathrm{Y}}^{(l)}\big)\Big)^2\Big]\Big)\,\mathrm{Sin}\big(\mathbf{W}_{\mathrm{g}}^{(l)}[x, y] + \mathrm{b}_{\mathrm{g}}^{(l)}\big)$$

$$\boldsymbol{\theta}^{(l)} = \Big\{\boldsymbol{\gamma}_{\mathrm{X}}^{(l)} \in \mathbb{R}^{\mathrm{N_{hid}}}, \boldsymbol{\gamma}_{\mathrm{Y}}^{(l)} \in \mathbb{R}^{\mathrm{N_{hid}}}, \boldsymbol{\mu}_{\mathrm{X}}^{(l)} \in \mathbb{R}^{\mathrm{N_{hid}}}, \boldsymbol{\mu}_{\mathrm{Y}}^{(l)} \in \mathbb{R}^{\mathrm{N_{hid}}}, \mathbf{W}_{\mathrm{g}}^{(l)} \in \mathbb{R}^{\mathrm{N_{hid}} \times 2}, \mathrm{b}_{\mathrm{g}}^{(l)} \in \mathbb{R}^{\mathrm{N_{hid}}}\Big\}$$

To analyse the maximum frequency component f^+_{MAGNet}, we analyse the frequency components of the Gabor filters used in MAGNet, and retain their maximum. We then plug the found frequency component into the analysis of Fathony *et al*. [1] to show how the frequency responses of Gabor filters and linear layers interact in MFNs. Finally, we add the effect of the FlexConv Gaussian mask to our analysis to obtain the maximum frequency component ot the final FlexConv kernel f^+_{FlexConv}.

Sine term in Gabor filters. In a Gabor filter, the sine term is multiplied with a Gaussian envelope. The frequency (in radians) of a sine function of the form $\mathrm{Sin}(\boldsymbol{w}^T[x, y] + b)$ is given by $\boldsymbol{w}$. We divide by 2π to convert the frequency units to Hertz, for compatibility with the rest of the analysis. For 2D inputs, the maximum frequency component of the sine function correspond to the largest frequency in the two input dimensions:

$$f^+_{\mathrm{Sin}} = \max_j \frac{w_j}{2\pi}. \tag{6.10}$$

The sine terms in MAGNets have multiple output channels: $\mathrm{Sin}\big(\mathbf{W}_{\mathrm{g}} \cdot [x, y] + \mathrm{b}_{\mathrm{g}}^{(l)}\big)$. Effectively, we compute the sine term independently for each channel:

$$f^+_{\mathrm{Sin},i} = \max_j \frac{\mathbf{W}_{\mathrm{g},i,j}}{2\pi}. \tag{6.11}$$

Gaussian term in Gabor filters. In a Gabor filter, a Gaussian envelope modulates a sine term. Let us assume for now that the Gaussian envelope is isotropic, rather than anisotropic as in MAGNets, and has single-channel output. By applying the convolution theorem, the sine term is equivalently convolved with

the Fourier transform of the Gaussian envelope in the frequency domain. Since the Fourier transform of a Gaussian envelope is another Gaussian envelope, the application of a Gaussian envelope amounts to blurring with a Gaussian kernel in the frequency domain. The size of the envelope in the Fourier domain σ_{F} can be derived from the standard deviation of the Guassian envelope in the spatial domain σ_{T} as follows:

$$\sigma_{\mathrm{T}}\sigma_{\mathrm{F}} = \frac{1}{2\pi} \Rightarrow \sigma_{\mathrm{F}} = \frac{1}{2\pi\sigma_{\mathrm{T}}}. \tag{6.12}$$

Gaussian blurs induce impulse signals to have a long tail. Consequently, we must define a cutoff point for this tail in terms of standard deviations to derive the maximum added frequency induced by the blur. We describe the cutoff point as $\sigma_{\mathrm{cut}} \in \mathbb{N}$. Typical choices for σ_{cut} are known as the *empirical*, or the "68-95-99.7" rule [2]. We choose a standard of two standard deviations, i.e., $\sigma_{\mathrm{cut}} = 2$, which covers 95% of the mass of the Gaussian envelope.

For an isotropic Gabor filter with $\gamma = \sigma_{\mathrm{T}}^{-1}$, the maximum frequency of its Gaussian envelope f_{env}^{+} is:

$$f_{\mathrm{env}}^{+} = \frac{\sigma_{\mathrm{cut}}}{2\pi(\sigma_{\mathrm{T}})^{-1}} = \frac{\sigma_{\mathrm{cut}}\gamma}{2\pi}. \tag{6.13}$$

Anisotropic envelopes. Our analysis so far assumes an isotropic Gaussian envelope in the Gabor filter. However, we need to account for the anisotropic Gaussian envelopes in MAGNets. Anisotropic filters have not one but two γ parameters: $\{\gamma_{\mathrm{X}}, \gamma_{\mathrm{Y}}\}$. The smallest of these will contribute most to f_{env}^{+}, as it will blur the most, so it is sufficient to compute f_{env}^{+} only using the smallest of the two γ terms:

$$f_{\mathrm{env}}^{+}(\gamma_X, \gamma_Y) = f_{\mathrm{env}}^{+}(\min\{\gamma_X, \gamma_Y\}). \tag{6.14}$$

The other assumption we made before was to work with single-channel outputs. MAGNets however use multi-channel outputs with independent Gaussian terms. The maximum frequency of multi-channel Gaussian envelopes is given by:

$$f_{\mathrm{env},i}^{+}(\boldsymbol{\gamma}_{\mathrm{X}}, \boldsymbol{\gamma}_{\mathrm{Y}}) = f_{\mathrm{env}}^{+}\left(\min\{\boldsymbol{\gamma}_{\mathrm{X},i}, \boldsymbol{\gamma}_{\mathrm{Y},i}\}\right) = \frac{\sigma_{\mathrm{cut}} \min\{\boldsymbol{\gamma}_{\mathrm{X},i}, \boldsymbol{\gamma}_{\mathrm{Y},i}\}}{2\pi}, \tag{6.15}$$

where the subscript i indexes the channels of the multi-channel Gaussian envelopes.

Maximum frequency component of anisotropic Gabor filters. Finally, the maximum frequency component of the i-th channel of an anisotropic Gabor filter

g is given by:

$$f^{+}_{\text{Gabor},i} = f^{+}_{\text{Sin},i}(\mathbf{W}_{\text{g}}) + f^{+}_{\text{env},i}(\boldsymbol{\gamma}_{\text{X}}, \boldsymbol{\gamma}_{\text{Y}}) \\ = \left(\max_{j} \frac{\mathbf{W}_{\text{g},i,j}}{2\pi}\right) + \frac{\sigma_{\text{cut}} \min\{\boldsymbol{\gamma}_{\text{X},i}, \boldsymbol{\gamma}_{\text{Y},i}\}}{2\pi}. \tag{6.16}$$

Fig. 6.A.1 illustrates the frequency spectrum of an example Gabor filter.

Maximum frequency component of a MAGNet. Fathony *et al*. [1] characterize the expansion of each term of the isotropic Gabor layers in MFNs in the final MFN output. In Eq. 25, Fathony *et al*. [1] demonstrate that the MFN representation contains a set of sine frequencies $\overline{\boldsymbol{\omega}}$ given by:

$$\overline{\boldsymbol{\omega}} = \left\{ s_{\text{L}} \omega^{(\text{L})}_{i_{\text{L}}} + s_{\text{L}-1} \omega^{(\text{L}-1)}_{i_{\text{L}-1}} + \cdots + s_l \omega^{(2)}_{i_2} + \omega^{(1)}_{i_1} \right\}. \tag{6.17}$$

Here, the indexes $i_1, i_2, \cdots, i_{\text{L}-1}$ range over all possible indices of each hidden unit of each layer of an MFN, and $s_2, \cdots, s_{\text{L}} \in \{-1, +1\}$ range over all $2^{\text{L}-1}$ possible binary signs. In other words, Fathony *et al*. [1] demonstrate that the representation of an MFN at a particular layer contains an exponential combination of all possible positive and negative combinations of the frequencies of the sine terms in each hidden unit at each layer in the MFN up to the current layer.

The original analysis uses these terms to argue that MFNs model exponentially many terms through a linear amount of layers. For our purpose of computing the frequency response of the MAGNet generated kernel, we can plug our derivation of the frequencies of the Gabor filter f_{Gabor} into $\overline{\boldsymbol{\omega}}$ to compute the frequency spectrum of the generated kernel:

$$\boldsymbol{f}^{+}_{\text{MAGNet}} = \left\{ s_{\text{L}} f^{(\text{L})}_{\text{Gabor},i_{\text{L}}} + s_{\text{L}-1} f^{(\text{L}-1)}_{\text{Gabor},i_{\text{L}-1}} + \cdots + s_2 f^{(2)}_{\text{Gabor},i_2} + f^{(1)}_{\text{Gabor},i_1} \right\} \tag{6.18}$$

As stated before, we are only interested in the maximum frequency in the frequency spectrum. We can therefore simplify Eq. 6.18 in two ways. First, we simplify over MAGNet layers by taking the maximum value of the spectrum, which is the sum over all layers using only the positive binary signs in s_{L} (Eq. 6.19). Next, we simplify over channel indices by retaining only the channel index that results in the highest frequency (Eq. 6.20). The maximum frequency of a

MAGNet is shown in Eq. 6.21:

$$\begin{aligned} \boldsymbol{f}^{+}_{\text{MAGNet}} &= \left\{(+1) f^{+\,(\text{L})}_{\text{Gabor},i_{\text{L}}} + (+1) f^{+\,(\text{L}-1)}_{\text{Gabor},i_{\text{L}-1}} + \cdots + (+1) f^{+\,(2)}_{\text{Gabor},i_2} + f^{+\,(1)}_{\text{Gabor},i_1}\right\} \qquad (6.19)\\ &= \left\{f^{+\,(\text{L})}_{\text{Gabor},i_{\text{L}}} + f^{+\,(\text{L}-1)}_{\text{Gabor},i_{\text{L}-1}} + \cdots + f^{+\,(2)}_{\text{Gabor},i_2} + f^{+\,(1)}_{\text{Gabor},i_1}\right\} \\ f^{+}_{\text{MAGNet}} &= \max_{i_{\text{L}}}\left(f^{+\,(\text{L})}_{\text{Gabor},i_{\text{L}}}\right) + \max_{i_{\text{L}-1}}\left(f^{+\,(\text{L}-1)}_{\text{Gabor},i_{\text{L}-1}}\right) \cdots + \max_{i_2}\left(f^{+\,(2)}_{\text{Gabor},i_2}\right) + \max_{i_1}\left(f^{+\,(1)}_{\text{Gabor},i_1}\right) \qquad (6.20)\\ &= \sum_{l=1}^{\text{L}} \max_{i_l}\left(f^{+\,(l)}_{\text{Gabor},i_l}\right) \\ &= \sum_{l=1}^{\text{L}} \max_{i_l}\left(\left(\max_j \frac{\mathbf{W}^{(l)}_{\text{g},i_l,j}}{2\pi}\right) + \frac{\sigma_{\text{cut}} \min\{\boldsymbol{\gamma}^{(l)}_{\text{X},i_l}, \boldsymbol{\gamma}^{(l)}_{\text{Y},i_l}\}}{2\pi}\right). \qquad (6.21) \end{aligned}$$

Effect of the Gaussian mask in the frequency components of a FlexConv. FlexConvs attenuate the MAGNet output with a Gaussian mask. The Gaussian mask (Eq. 6.1) works analogously to the Gaussian envelope term in the Gabor filter: it blurs the frequency components of the generated kernel with standard deviation σ_{F}. Therefore, we can reuse our derivation for the Gaussian envelope of the Gabor filter (Eq. 6.15). The maximum frequency component of a FlexConv kernel is given by:

$$\begin{aligned} f^{+}_{\text{FlexConv}} &= f^{+}_{\text{MAGNet}} + f^{+}_{\text{env}} \\ &= f^{+}_{\text{MAGNet}} + \frac{\sigma_{\text{cut}} \min\{\sigma_{\text{X}}^{-1}, \sigma_{\text{Y}}^{-1}\}}{2\pi} = f^{+}_{\text{MAGNet}} + \frac{\sigma_{\text{cut}}}{\max\{\sigma_{\text{X}}, \sigma_{\text{Y}}\} 2\pi} \\ &= \sum_{l=1}^{\text{L}} \max_{i_l}\left(\left(\max_j \frac{\mathbf{W}^{(l)}_{\text{g},i_l,j}}{2\pi}\right) + \frac{\sigma_{\text{cut}} \min\{\boldsymbol{\gamma}^{(l)}_{\text{X},i_l}, \boldsymbol{\gamma}^{(l)}_{\text{Y},i_l}\}}{2\pi}\right) + \frac{\sigma_{\text{cut}}}{\max\{\sigma_{\text{X}}, \sigma_{\text{Y}}\} 2\pi}. \qquad (6.22) \end{aligned}$$

Visualization of regularized kernels. Fig. 6..8 shows example kernels from FlexNets trained with aliasing regularization. The frequency domain plots confirm the accuracy of our frequency component regularization.

6.A.2 Regularizing the frequency response of FlexConv

Nyquist frequency of a FlexConv kernel. Given the sampling rate f_s of the kernel, we can compute its Nyquist frequency f_{Nyq} as:

$$f_{\text{Nyq}} = \frac{1}{2} f_s \tag{6.23}$$

To compute the sampling rate, we note that the kernel coordinates input to our MAGNet stretch over a $[-1, 1]^D$ domain. For a kernel of length k, we therefore sample one point in every $f_s = \frac{k-1}{2}$ units.

Knowing the sampling rate in terms of the kernel size allows us to express the Nyquist frequency in terms of the (pre-masked) kernel size:

$$f_{\text{Nyq}}(k) = \frac{1}{2} \frac{k-1}{2} = \frac{k-1}{4}. \tag{6.24}$$

Note that the kernel size in a FlexConv is initialized to be equal to the resolution of the data, if it is odd. For even resolutions, it corresponds to the resolution of the data plus one.

Constructing the regularization term. We train FlexConv with a regularization term on the frequency response of the generated kernel to ensure that aliasing effects do not distort the performance of the model when it is inferred at a higher resolution. This section details the implementation of the regularization function.

From the parameters of each FlexConv module, we compute f^+_{FlexConv} according to Eq. 6.22. For the amount of standard deviations to use in determining f^+_{env} (Eq. 6.15) we use $h = 2$. From the kernel size k of the FlexConv module we compute $f_{\text{Nyq}}(k)$ according to Eq. 6.24. We then apply an L2 regularizer over the amount that f^+_{FlexConv} exceeds $f_{\text{Nyq}}(k)$:

$$\mathcal{L}_{\text{HF}} = ||\max\{f^+_{\text{FlexConv}}, f_{\text{Nyq}}(k)\} - f_{\text{Nyq}}(k)||^2. \tag{6.25}$$

We weight $\mathcal{L}_{\text{HF}}$ by $\lambda = 0.1$ when adding it to our loss function.

Improved implementation. Eq. 6.25 contains a sum over the L layers of the MAGNet. In practice, we prefer to regularize each layer $l \in \text{L}$ separately, so that the gradients of the regularization of different layers are not dependent on

each other. We therefore implement the anti-aliasing regularization by regularizing each MAGNet layer independently, and spreading the f^+_{env} term from the gaussian mask uniformly over all MAGNet layers:

$$\mathscr{L}_{\text{HF},l} = ||\max\left\{f^+_{\text{MAGNet},l} + \frac{f^+_{\text{env}}}{\text{L}}, \frac{f_{\text{Nyq}}(k)}{\text{L}}\right\} - \frac{f_{\text{Nyq}}(k)}{\text{L}}||^2 \tag{6.26}$$

$$= ||\max\left\{\max_{i_l}\left(f^{+\,(l)}_{\text{Gabor},i_l}\right) + \frac{f^+_{\text{env}}}{\text{L}}, \frac{f_{\text{Nyq}}(k)}{\text{L}}\right\} - \frac{f_{\text{Nyq}}(k)}{\text{L}}||^2. \tag{6.27}$$

In the code, we refer to this method as the `together` method, versus the `summed` method of Eq. 6.25. In preliminary experiments, we observed improved performance of anti-aliasing training when using the `together` method. All of our experiments anti-aliasing experiments therefore use the `together` setting.

6.B Dataset Description

6.B.1 Image Fitting Datasets

Kodak dataset. The Kodak dataset [3] consists of 24 natural images of size 768×512. This dataset is a popular benchmark used for compression and image fitting methods.

6.B.2 Sequential Datasets

Sequential and Permuted MNIST. The sequential MNIST dataset (sMNIST) [4] takes the 28×28 images from the original MNIST dataset [5], and presents them as a sequence of 784 pixels. The goal of this task is to perform digit classification given the representation of the last sequence element of a sequential model. Consequently, good predictions require the model to preserve long-term dependencies up to 784 steps in the past.

The permuted MNIST dataset (pMNIST) additionally changes the order of all the sMNIST sequences by a random permutation. Consequently, models can no longer rely on local features to construct good feature representations. As a result, the classification problem becomes more difficult, and the importance of long-term dependencies more pronounced.

Sequential and Noise-Padded CIFAR10. The sequential CIFAR10 dataset (sCIFAR10) [6] takes the 32×32 images from the original CIFAR10 dataset [7] and

presents them as a sequence of 1,024 pixels. The goal of this task is to perform image classification given the representation of the last sequence element of a sequential model. This task is more difficult than sMNIST, as a larger memory horizon is required to solve the task and more complex structures and intra-class variations are present in the data [8].

The noise-padded CIFAR10 dataset (npCIFAR10) [9] flattens the images from the original CIFAR10 dataset [7] along their rows to create a sequence of length 32, and 96 channels (32 rows × 3 channels). Next, these sequences are concatenated with 968 entries of noise to form the final sequences of length 1000. As for sCIFAR10, the goal of the task is to perform image classification given the representation of the last sequence element of a sequential model.

CharacterTrajectories. The CharacterTrajectories dataset is part of the UEA time series classification archive [10]. It consists of 2858 time series of length 182 and 3 channels representing the x, y positions, and the tip force of a pen while writing Latin alphabet characters in a single stroke. The goal is to classify out of 20 classes the written character using the time series data.

Speech Commands. The Speech Commands dataset [11] consists of 105,809 one-second audio recordings of 35 spoken words sampled at 16kHz. Following [12], we extract 34975 recordings from ten spoken words to construct a balanced classification problem. We refer to this dataset as **SpeechCommands_raw**, or **SC_raw** for short. Furhtermore, we utilize the preprocessing steps of [12] and extract mel-frequency cepstrum coefficients from the raw data. The resulting dataset, abreviated **SC**, consists of time series of length 101, and 20 channels.

6.B.3 Image Benchmark Datasets

MNIST. The MNIST hadwritten digits datset [13] consists of 70,000 gray-scale handwritten digits of size 28×28, divided into a training and test sets of 60,000 and 10,000 images, respectively. The goal of the task is to classify these digits as one of the ten possible digits $(0, 1, ..8, 9)$.

CIFAR-10 The CIFAR-10 dataset [7] consists of 60,000 natural images from 10 classes of size 32×32, divided into training and test sets of 50,000 and 10,000 images, respectively.

STL-10. The STL-10 dataset [14] is a subset of the ImageNet dataset [15] consisting of 5,000 natural images from 10 classes of size 96×96, divided into trainint and test sets of 4,500 and 500 images, respectively.

ImageNet-k. The Imagenet-k [16] dataset is derived from the ImageNet dataset [17] by downsampling all samples to a resolution $k \in [64, 32, 16, 8]$. The dataset contains 1000 classes with 1,281,167 training samples and 50,000 validation

samples.

6.C Additional Experiments

Table 6.C.1: Average PSNR for fitting of images in the Kodak dataset. Both our improved initialization scheme, as well as the inclusion of anisotropic Gabor functions lead to better reconstructions.

Model	# Params	Improved Init	PSNR
SIREN	7.14K	-	25.665
$\text{MFN}_{\text{Fourier}}$	7.40K	-	23.276
$\text{MFN}_{\text{Gabor}}$	7.11K	✗	25.361
		✓	**25.606**
MAGNet	7.36K	✗	25.791
		✓	**25.893**

6.C.1 Image Classification

CIFAR-10. Table 6.C.2 shows all results for our CIFAR-10 experiments, including more ablations.

ImageNet-32. Results for the ImageNet-32 experiment are shown in Table 6.C.3. FlexNets are slightly worse than CIFARResNet-32 [20] with slightly less parameters. However, the results reported by [16] for Wide ResNets [21] outperform FlexNets by a significant margin.

Alias-free ImageNet-32. We report results for alias-free FlexNets on ImageNet-k [16] in Table 6.C.4, to verify the results of alias-free training at a larger scale. We find that FlexConv and N-Jet both mostly retain classification accuracy between source and target resolution, while CIFARResNet-32 degrades drastically.

MNIST and STL-10. We additionally report results on MNIST (Table 6.C.5) and STL-10 (Table 6.C.6). We choose these dataset for the difference in image sizes of the training data. On MNIST, though performance on MNIST is quite saturated, we are competitive with state of the art methods. On STL-10 we are significantly worse than the baseline CIFARResNet from Luo *et al*. [22], though with significantly less parameters. We were not able to prepare a more relevant baseline for this experiment.

Table 6.C.2: Full results on CIFAR-10. We report results over three runs per setting. CIFARResNet-44 w/ CKConv is a CIFARResNet-44 where all convolutional layers are replaced with CKConvs with $k = 3$. CIFARResNet-44 w/ FlexConv is a CIFARResNet-44 where all convolutional layers are replaced with FlexConv with learned kernel size, except for the shortcut connections of the strided convolutional layers, which are pointwise convolutions. *Results are taken from the respective original works instead of reproduced. † Results are from single run.

Model	Size	CIFAR-10 Acc.
DCN-σ^{ji} [18]	0.47M	89.7 ± 0.3*
N-Jet-CIFARResNet32 [19]	0.52M	92.3 ± 0.3*
N-Jet-ALLCNN [19]	1.07M	92.5 ± 0.1*
CIFARResNet-44 [20]	0.66M	92.9*†
CIFARResNet-44 [20] (our reproduction)	0.66M	90.9 ± 0.2
CIFARResNet-44 w/ CKConv ($k = 3$)	2.58M	86.1 ± 0.9
CIFARResNet-44 w/ FlexConv	2.58M	81.6 ± 0.8
FlexNet-7 w/ conv. ($k = 3$)	0.17M	89.5 ± 0.3
FlexNet-7 w/ conv. ($k = 33$)	20.0M	78.0 ± 0.3
FlexNet-7 w/ N-Jet [19]	0.70M	91.7 ± 0.1
$\text{CKCNN}_{\text{SIREN}}$-3	0.26M	72.4*
$\text{CKCNN}_{\text{Fourier}}$-3	0.27M	83.8*
$\text{CKCNN}_{\text{Gabor}}$-3	0.28M	85.6*
$\text{CKCNN}_{\text{MAGNet}}$-3	0.28M	86.2*
CKCNN-7	0.63M	71.7*
$\text{CKCNN}_{\text{Fourier}}$-7	0.63M	84.6*
$\text{CKCNN}_{\text{Gabor}}$-7	0.67M	87.7*
$\text{CKCNN}_{\text{MAGNet}}$-7	0.67M	85.9*
$\text{FlexNet}_{\text{SIREN}}$-7	0.63M	88.9*
$\text{FlexNet}_{\text{Fourier}}$-7	0.66M	91.6*
$\text{FlexNet}_{\text{Gabor}}$-7	0.67M	92.0*
FlexNet-3	0.27M	90.4 ± 0.2
FlexNet-5	0.44M	91.0 ± 0.5
FlexNet-7	0.67M	92.2 ± 0.1

Table 6.C.3: Results on ImageNet-32. *Results are taken from the respective original works instead of reproduced. † Results are from a single run.

Model	Size	ImageNet-32 Top-1	ImageNet-32 Top-5
CIFARResNet-32	0.53M	26.41 ± 0.13	49.37 ± 0.15
WRN-28-1	0.44M	32.03*†	57.51*†
FlexNet-5	0.44M	24.9 ± 0.4	47.7 ± 0.6

Table 6.C.4: Results for alias-free FlexNets on CIFAR-10 and ImageNet-k. Δ denotes difference in accuracy.

Model	Size	ImageNet-k Top-1 $k = 16$	$\Delta_{k=16}$ $k = 32$
CIFARResNet-32	0.52M	16.1 ± 0.0	-11.6 ± 0.4
FlexNet-5 w/ N-Jets	0.46M	15.7 ± 0.1	-1.9 ± 0.4
FlexNet-5	0.44M	14.9 ± 0.1	-1.9 ± 1.7

Table 6.C.5: Results on MNIST. We train each model with three different seeds and report mean and standard deviation. *Results are taken from the respective original works instead of reproduced. † Results are from single run.

Model	Size	MNIST Acc.
Efficient-CapsNet [23]	0.16M	**99.8***†
Network in Network [24]	N/A	99.6*†
VGG-5 (results from [25])	3.65M	99.7*†
FlexNet-16	0.67M	99.7 ± 0.0

6.D Experimental Details

6.D.1 FlexNet

We propose an image classification architecture named *FlexNet* (Fig. 6.D.1), consisting of a stack of FlexConv blocks followed by a global average pooling layer

Table 6.C.6: Results on STL-10. We train each model with three different seeds and report mean and standard deviation. *Results are taken from [22]. † Results are from single run.

Model	Size	STL-10 Acc.
CIFARResNet-18	11.2M	81.0*†
FlexNet-16	0.67M	68.6 ± 0.7

and a linear layer. FlexNets are named "FlexNet-L" where L indicates the amount of layers in the architecture.

FlexBlock. Each FlexBlock consists of two FlexConvs with BatchNorm [26] and dropout [27] ($d = 0.2$) as well as a residual connection. The width of a block i is determined by scaling a base amount c by progressively increasing factors: $c_i = [c, c \times 1.5, c \times 1.5, c \times 2.0, c \times 2.0](i)$. The default configuration of FlexNet uses $c = 22$. In FlexNet-N-Jet models, we scale c to match the amount of parameters of the FlexNet in the comparison.

FlexConv initialization. We initialize the FlexConv mask variances small, at $\sigma_X^2, \sigma_Y^2 = 0.125$. For initializing MAGNet, we initialize the Gaussian envelopes as discussed in Section 6.3.2. We initialize the linear layer weights by the same Gamma distribution as used for the enveloped, modulated by a scaling factor of 25.6. We found that this value of the scaling factor, rather than a higher one, helped in reducing the performance of alias-free models. We initialize the bias of the linear layers by $\mathcal{U}(-\pi, \pi)$.

CIFAR-10. In FlexNet-16 models for CIFAR-10 we use $c = 24$ to approximate the parameter count of CIFARResNets in the experiment.

6.D.2 Optimization

We use Adam [28] to optimize FlexNet. Unless otherwise specified, we use a learning rate of 0.01 with a cosine annealing scheme [29] with five warmup epochs. We use a different learning rate of 0.1× the regular learning rate for the FlexConv Gaussian mask parameters. We do not use weight decay, unless otherwise specified.

Kodak. We overfit on each image of the dataset for 20,000 iterations. To this end, we use a learning rate of 0.01 without any learning rate scheme. We observe that SIRENs diverge with this learning rate and thus, reduce the learning rate to

0.001 for these models.

CIFAR-10. We train for 350 epochs with a batch size of 64. We use the data augmentation from He *et al*. [20] when training CIFAR-10: a four pixel padding, followed by a random 32 pixel crop and a random horizontal flip.

ImageNet-32. We train for 350 epochs with a batch size of 2048. We use the same data augmentation as used for CIFAR-10. We do use a weight decay of 1e−5 for ImageNet-32 training.

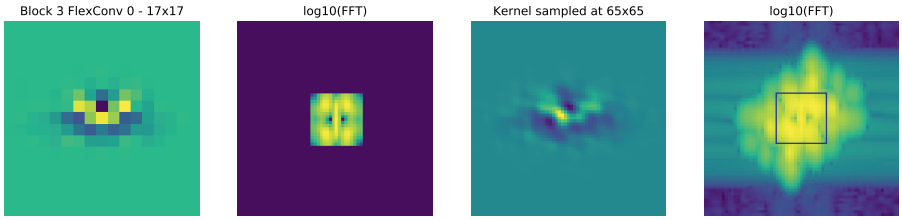
Sequential and Permuted MNIST. We train for 200 epochs with a batch size of 64 and a learning rate of 0.01. We use a weight decay of 1e−5.

Sequential and Noise-Padded CIFAR-10. For sequential CIFAR-10, we train for 200 epochs with a batch size of 64, a learning rate of 0.001 and a weight decay of 1e−5. For noise-padded CIFAR-10, we train for 300 epochs with a batch size of 32, a learning rate of 0.01 and no weight decay.

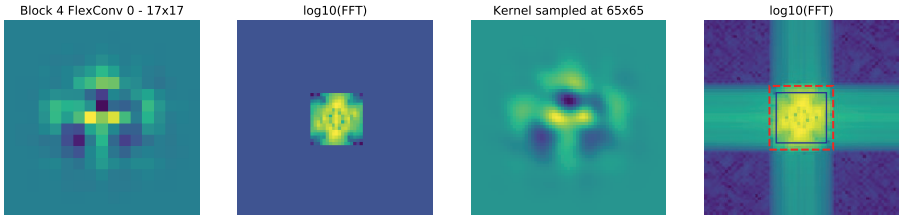
Speech Commands and CharTrajectories. We train for 300 epochs with a batch size of 32 and a learning rate of 0.001. For CharTrajectories, we use a weight decay of 1e−5.

6.D.3 Rotated Gaussian masks

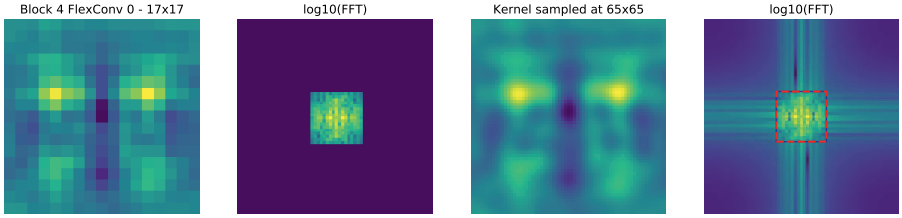
MAGNets use anisotropic Gaussian terms in the Gabor filters, which yields improvements in descriptive power and convergence speed (Section 6.3.2). For the same reason, we explore making the anisotropic FlexConv Gaussian mask steerable, by including an additional vector of learnable angle parameters $\boldsymbol{\phi}^{(l)} \in \mathbb{R}^{N_{\text{hid}}}$ that rotates the Gaussian masks. Although preliminary experiments show rotated masks lead to slight additional improvements, the computational overhead required to rotate the masks is large. Consequently, we do not consider rotated Gaussian masks in our final experiments.



(a) No regularization, block 3 of 7.



(b) Regularizing $f^{+}_{\text{MAGNet}'}$, block 4 of 7.



(c) Regularizing $f^{+}_{\text{FlexConv}'}$, block 4 of 7.

Figure 6..8: Example kernels from FlexNet-16 models trained (i) without regularization, (ii) with aliasing regularization of $f^{+}_{\text{MAGNet}'}$, (iii) with aliasing regularization of $f^{+}_{\text{FlexConv}'}$. In the columns, from left to right: (i) original kernel at 33×33, (ii) FFT of the original kernel, (iii) kernel inferred at 65×65, to find aliasing effects, (iiii) FFT of the 65×65 kernel, with the solid line showing the Nyquist frequency of the 33×33 kernel, and the red dotted line showing the maximum frequency component as computed by our analysis. For $f^{+}_{\text{FlexConv}'}$ the maximum frequency matches almost exactly with the Nyquist frequency, showing that our aliasing regularization works. For $f^{+}_{\text{MAGNet}'}$, the maximum frequency is slightly higher than the Nyquist frequency, as the FlexConv mask is not included in the frequency term derivation. This is reflected in the slightly worse resolution generalization results reported in Section 6.4.3. Furthermore, some aliasing effects are still apparent for the aliasing regularized models, as discussed in Section 6.6.

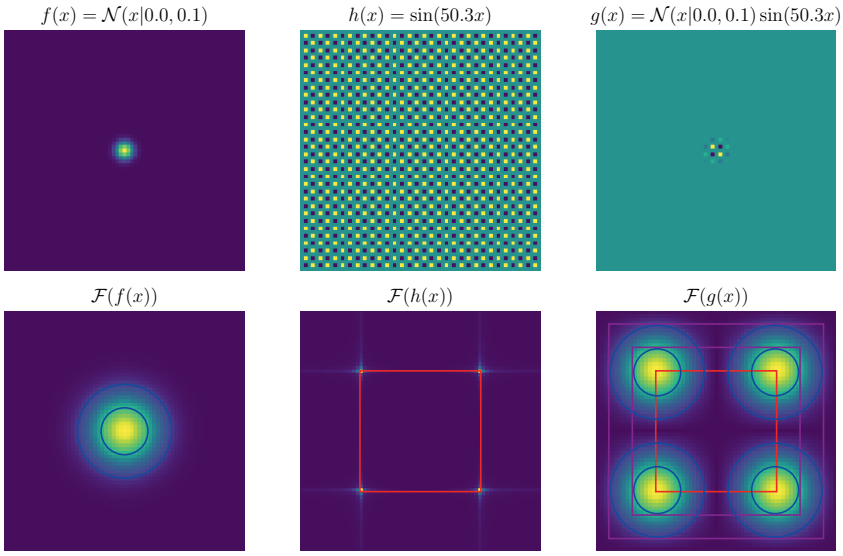


Figure 6.A.1: Decomposition of a Gabor filter and its frequency spectrum. Top row: a decomposition of a Gabor filter (right) into its Gaussian term (left) and its sine term (center). Bottom row: frequency responses for each respective filter. The Fourier transform of a Gaussian envelope is a Gaussian envelope (blue circles show $\sigma_{\mathscr{F}}$ for $h = \{1, 2\}$). The Fourier transform of a sine pattern is a collection of symmetrical impulse signals (red box shows the Nyquist frequency). The Gaussian envelope blurs the frequency response of the sine term (purple boxes show the frequency response for $h = \{1, 2, 3\}$).

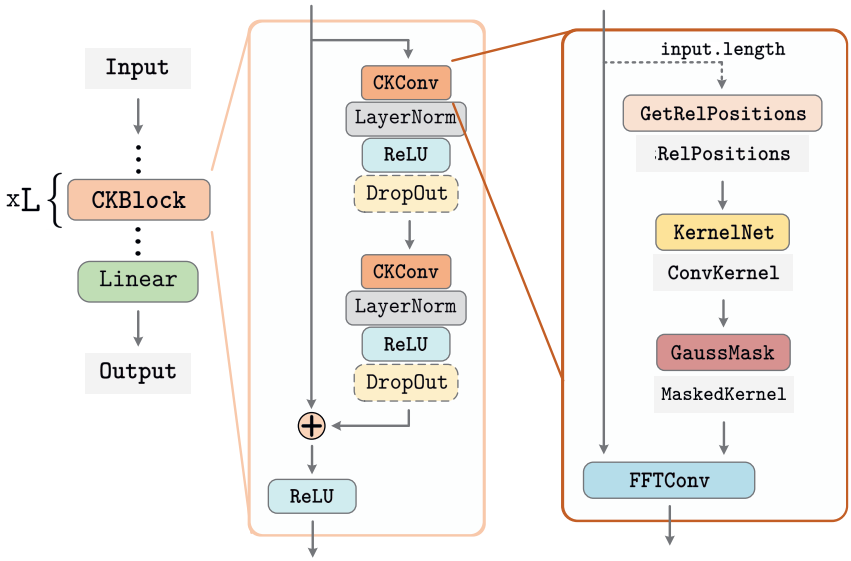


Figure 6.D.1: FlexNet architecture. FlexNet-L consists of L FlexBlocks, where each FlexBlock is a residual block of FlexConvs.

REFERENCES

[1] R. Fathony, A. K. Sahu, D. Willmott, and J. Z. Kolter. "Multiplicative Filter Networks". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=OmtmcPkkhT.

[2] A. Hald. "De Moivre's Normal Approximation to the Binomial, 1733, and Its Generalization". In: *A History of Parametric Statistical Inference from Bernoulli to Fisher, 1713–1935* (2007), pp. 17–24.

[3] Kodak. *Kodak Dataset*. 1991. URL: http://r0k.us/graphics/kodak/.

[4] Q. V. Le, N. Jaitly, and G. E. Hinton. "A simple way to initialize recurrent networks of rectified linear units". In: *arXiv preprint arXiv:1504.00941* (2015).

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[6] S. Chang, Y. Zhang, W. Han, M. Yu, X. Guo, W. Tan, X. Cui, M. Witbrock, M. A. Hasegawa-Johnson, and T. S. Huang. "Dilated recurrent neural networks". In: *Advances in neural information processing systems*. 2017, pp. 77–87.

[7] A. Krizhevsky, G. Hinton, *et al. Learning multiple layers of features from tiny images*. Tech. rep. 2009.

[8] S. Bai, J. Z. Kolter, and V. Koltun. "Trellis networks for sequence modeling". In: *arXiv preprint arXiv:1810.06682* (2018).

[9] B. Chang, M. Chen, E. Haber, and E. H. Chi. "AntisymmetricRNN: A dynamical system view on recurrent neural networks". In: *arXiv preprint arXiv:1902.09689* (2019).

[10] A. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh. "The UEA multivariate time series classification archive, 2018". In: *arXiv preprint arXiv:1811.00075* (2018).

[11] P. Warden. "Speech commands: A dataset for limited-vocabulary speech recognition". In: *arXiv preprint arXiv:1804.03209* (2018).

[12] P. Kidger, J. Morrill, J. Foster, and T. Lyons. "Neural controlled differential equations for irregular time series". In: *arXiv preprint arXiv:2005.08926* (2020).

[13] Y. LeCun and C. Cortes. "MNIST handwritten digit database". In: (2010). URL: http://yann.lecun.com/exdb/mnist/.

[14] A. Coates, A. Ng, and H. Lee. "An analysis of single-layer networks in unsupervised feature learning". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 215–223.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[16] P. Chrabaszcz, I. Loshchilov, and F. Hutter. "A Downsampled Variant of ImageNet as an Alternative to the CIFAR datasets". In: *CoRR* abs/1707.08819 (2017). arXiv: 1707.08819. URL: http://arxiv.org/abs/1707.08819.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[18] N. Tomen, S.-L. Pintea, and J. Van Gemert. "Deep Continuous Networks". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 10324–10335. URL: https://proceedings.mlr.press/v139/tomen21a.html.

[19] S. L. Pintea, N. Tomen, S. F. Goes, M. Loog, and J. C. van Gemert. "Resolution learning in deep convolutional networks using scale-space theory". In: *arXiv preprint arXiv:2106.03412* (2021).

[20] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[21] S. Zagoruyko and N. Komodakis. "Wide residual networks". In: *arXiv preprint arXiv:1605.07146* (2016).

[22] C. Luo, J. Zhan, L. Wang, and W. Gao. "Extended batch normalization". In: *arXiv preprint arXiv:2003.05569* (2020).

[23] V. Mazzia, F. Salvetti, and M. Chiaberge. "Efficient-CapsNet: Capsule Network with Self-Attention Routing". In: *arXiv preprint arXiv:2101.12491* (2021).

[24] M. Lin, Q. Chen, and S. Yan. "Network in network". In: *arXiv preprint arXiv:1312.4400* (2013).

[25] H. Kabir, M. Abdar, S. M. J. Jalali, A. Khosravi, A. F. Atiya, S. Nahavandi, and D. Srinivasan. "Spinalnet: Deep neural network with gradual input". In: *arXiv preprint arXiv:2007.03347* (2020).

[26] S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[28] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[29] I. Loshchilov and F. Hutter. "Sgdr: Stochastic gradient descent with warm restarts". In: *arXiv preprint arXiv:1608.03983* (2016).

7

Discussion



This thesis explores how vision models learn to use spatial dataset biases, and how we can improve the usage of spatial biases using (flexible) inductive biases to benefit vision models. The previous chapters have shown that (i) inductive biases make models better in data-deficient settings (Chapter 2), (ii) how to measure to which degree vision models learn to use spatial biases (Chapters 3, 4) and (iii) how to integrate flexible inductive spatial biases to improve models (Chapters 5, 6). Several common themes arise that we discuss in the following.

7.1 Position bias

The translation invariance of Vision Transformers yields a unique framework to analyze dataset position bias: as appearance information and position information are cleanly separated between image tokens and position embeddings, we are able to measure the usage of position bias in ViT models directly (Chapter 4). We use this opportunity to answer the question whether vision models should learn not to use dataset position bias (low model position bias, i.e. high translation invariance) or whether they should (high model position bias). Our findings indicate it depends strongly on the dataset.

For photographic datasets such as ImageNet [1], we find that vision models learn low model position bias in intermediate representations (Chapter 3), but high model position bias in the overall network response, which benefits accuracy (Chapter 4). We hypothesize that early layers, which have been shown to recognize mostly low-level object parts [2, 3], benefit from sharing pattern recognition functions across positions, while later layers, which recognize objects as a whole in a semantic context, benefit from knowing the position of the object.

Our analysis further identifies key network design and training factors that modulate the extent of learned position bias. Chapters 3, 4 find that using data augmentation decreases learned model position bias, and consequently hurts performance on datasets with high position bias. Chapter 3 shows that the choice of architecture matters: ViTs learn higher model position bias than CNNs. In ViTs, the choice of position embedding also affects the learned model position bias (Chapter 4). All these design choices matter when trying to optimize the training of the vision model for a particular dataset position bias, so as to improve the model.

7.2 SCALE BIAS

Scale bias arises when the size of objects in an image varies systematically, either due to changes in distance from the camera or the composition of the dataset. We observe that scale equivariant methods are often not tuned very well to the scales apparent in vision datasets (Chapter 5). We therefore add flexible inductive bias that allows the model to fit to the scale distribution in the training dataset. Similarly, in Chapter 6 we make CNNs learn their kernel sizes, allowing the model to learn the appropriate receptive field for the scale distribution apparent in the training data. We find that the receptive fields learned by these CNNs match those of successful CNN designs [4, 5], indicating that architectures that arose from collective tuning by the research field capture the right receptive field.

Beyond tuning receptive fields, we also explore generalization across scales using the MAGNet parameterization of convolutional kernels. This approach allows kernels to transfer across different scales with minimal performance loss. Given the high cost of acquiring large labeled datasets, this is particularly valuable: a model pre-trained on a large-scale dataset can be efficiently adapted to a new dataset with a distinct scale distribution, thereby improving performance in data-limited settings. In this way, we contribute a method to make transfer learning more widely usable to improve models in data-deficient settings.

Overall, these contributions emphasize the importance of adaptive scale modeling in vision architectures. By enabling models to flexibly capture and exploit scale distributions, we enhance both data efficiency and generalization, providing a framework for robust application of vision models across heterogeneous datasets and diverse real-world scenarios.

7.3 FLEXIBLE INDUCTIVE BIAS

The concept of flexible inductive bias emerges as a unifying principle to adapt vision models to the spatial biases in different datasets. Across several chapters, we show the benefits of flexible inductive bias: in Chapter 4, we learn the degree to which to use dataset position bias using the γ parameter in Auto-PE; in Chapter 5, we learn the extent of the scale range using the learnable γ in ISR[1]; in Chapter 6, we learn the receptive field of the FlexNet by learning the anisotropic window size of the FlexConv. In each case, we show that making the network adapt to apparent dataset spatial biases specifically using our introduced flexible inductive bias, makes the model fit the training data better.

[1] In retrospect, we could have been a bit more creative in our choice of parameter notation...

We hypothesize that this effectiveness arises from isolating the learning of spatial biases into a small set of dedicated parameters. By decoupling these biases from the broader network parameters, the model can efficiently encode positional or scale-related patterns without requiring complex coordination across many layers, which would otherwise make bias learning less efficient and more error-prone.

Despite these advantages, there are downsides to incorporating (flexible) inductive spatial biases: they often incurs higher compute costs (Chapters 5, 6) [6–9], and their design is sensitive to mistakes: if the inductive bias is not correct, there is no performance advantage and only increased compute cost remains [10–12]. Nevertheless, when carefully applied, flexible inductive biases provide a principled and adaptable framework for improving data efficiency and robustness, enabling vision models to leverage spatial patterns without rigidly constraining their representations.

7.4 Methods for data-deficient settings

Data-deficient scenarios, such as those featured in our series of VIPriors challenges (Chapter 2), are a significant challenge for vision models: there is still a 15% gap in vision model performance between data-deficient and data-abundant settings. In our challenge, easy to use methods such as model ensembling and data augmentation are the most used methods to improve models, highlighting the continued utility of simple, well-understood techniques in data-deficient scenarios.

A central question is whether using prior knowledge, by way of inductive bias, helps in these settings. The challenge encouraged participants to explore the development of new inductive biases, which was recognized through a jury prize for innovative contributions. Nevertheless, only a small fraction of submissions proposed genuinely novel inductive biases. Among these, some achieved notable performance improvements. However, it is important to consider potential selection biases: participants are unlikely to submit methods that fail, and the time and effort required to design and implement new inductive biases may have deterred broader experimentation. In contrast, many participants leveraged existing architectures, augmentation strategies, loss functions, and other mechanisms that implicitly encode inductive biases, even if not formally novel.

From these observations, we infer that while inductive biases can indeed enhance model performance in data-deficient settings, the development of new biases entails substantial implementation costs. In practice, applying established

methods with proven efficacy may offer a more efficient allocation of development effort, particularly when immediate performance gains are the primary objective. These findings underscore a pragmatic balance in data-limited contexts: leveraging existing inductive biases judiciously can provide meaningful improvements, while the creation of new biases should be weighed against the associated resource investment.

7.5 FUTURE OUTLOOK

Benchmarking data-efficiency. In our VIPriors challenges (Chapter 2) we measured performance in a data-deficient setting. This is different from measuring data-efficiency directly, since the latter would measure the *change* in performance when removing data. While data augmentations and model ensembles may boost performance at any data scale, inductive biases theoretically have an advantage in data-efficiency [13]. Future challenges may incorporate proposed measures of data-efficiency, such as those by Hoiem *et al*. [14] or based on the work of Viering *et al*. [15], to better spotlight methods specifically addressing data-efficiency. Furthermore, combining the benchmarking of data-efficiency with compute-efficiency may counteract the success of inductive biases that are prohibitively more expensive and therefore not feasible in practice.

New applications. Our proposed MAGNet parameterization of FlexConv operators (Chapter 6) allows for the generalization of convolutions to unseen scales. Training models with MAGNet kernels allows engineers to transfer a model trained on a particular scale distribution, to another setting with a different known scale distribution. For example, to detect small objects, one could train a FlexNet with MAGNet kernels on large objects, and then change the scale of the MAGNet kernels to work on small objects. This may make it easier to train object detectors to work well on small objects, broadening the range of feasible real-world applications.

Spatial biases in large datasets. Classical machine learning theory suggests that inductive biases are useful for small datasets, but hinder models trained on large datasets [13]. Even though we show that inductive spatial biases are useful for vision models on current datasets, we wonder if incorporating inductive spatial biases will still be beneficial when datasets get larger.

Our proposed P-SHAP and learned equivariance measures (Chapters 3, 4) offer the tools to allow such analysis, though large-scale compute is necessary

to carry out these experiments. We hypothesize that position bias and scale bias are likely to persist in large datasets, as the mechanisms that generate these dataset biases (e.g. photography bias) are likely retained at larger data scales. Consequently, we hypothesize that using these dataset biases in vision models will remain important, even when datasets become ever bigger.

Dataset biases in other modalities. As the integration of tech products in society evolves, so does the modality of the datasets that vision models can be trained on. Using video instead of images as a data source is an active area of research. Furthermore, LLM-powered chatbots are increasingly replacing traditional internet search, creating large datasets of personal conversations, mixed with user-uploaded images and videos. As the abundance of data will move to other modalities, so will the biases apparent in datasets. Consequently, the design of model architectures and the incorporation of appropriate inductive biases must adapt to the unique characteristics of these new data modalities. Research into modality-specific inductive biases will be critical for developing AI systems that are both data-efficient and robust.

7.6 Final words

There are those who advocate against spending much time on inductive biases. The "bitter lesson" of Richard Sutton [16] prescribes that we should just add more data, not more inductive bias. However, data is finite and may become scarce sooner rather than later [17–19]. Moreover, raw performance of vision models, given as much data as possible, should not be our only goal: data-deficient settings are real, plenty and important. This thesis shows that spatial biases exist in the data that vision models are trained on, and that incorporating inductive spatial biases improves the data-efficiency of vision models. Therefore, we hypothesize that data-efficient vision models are the future of our field, and the search for appropriate inductive biases will remain an important endeavor.

REFERENCES

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[2] M. D. Zeiler and R. Fergus. "Visualizing and understanding convolutional networks". In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*. Springer. 2014, pp. 818–833.

[3] C. Olah, A. Mordvintsev, and L. Schubert. "Feature Visualization". In: *Distill* (2017). https://distill.pub/2017/feature-visualization. DOI: 10.23915/distill.00007.

[4] K. Simonyan and A. Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[5] K. He, X. Zhang, S. Ren, and J. Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[6] T. S. Cohen and M. Welling. "Group Equivariant Convolutional Networks". In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16. New York, NY, USA: JMLR.org, 2016, pp. 2990–2999.

[7] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow. "Harmonic networks: Deep translation and rotation equivariance". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5028–5037.

[8] I. Sosnovik, A. Moskalev, and A. Smeulders. "DISCO: accurate discrete scale convolutions". In: *Proceedings of the 32nd British Machine Vision Conference (BMVC)*. 2021.

[9] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. "Feature pyramid networks for object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[10] O. S. Kayhan and J. C. v. Gemert. "On translation invariance in cnns: Convolutional layers can exploit absolute spatial location". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 14274–14285.

[11] R. Zhang. "Making convolutional networks shift-invariant again". In: *International conference on machine learning*. PMLR. 2019, pp. 7324–7334.

[12] I. Sosnovik, A. Moskalev, and A. Smeulders. "How to Transform Kernels for Scale-Convolutions". en. In: *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. Montreal, BC, Canada: IEEE, Oct. 2021, pp. 1092–1097. ISBN: 978-1-66540-191-3. DOI: 10.1109/ICCVW54120.2021.00128. URL: https://ieeexplore.ieee.org/document/9607479/ (visited on 08/12/2022).

[13] U. Von Luxburg and B. Schölkopf. "Statistical learning theory: Models, concepts, and results". In: *Handbook of the History of Logic*. Vol. 10. Elsevier, 2011, pp. 651–706.

[14] D. Hoiem, T. Gupta, Z. Li, and M. Shlapentokh-Rothman. “Learning curves for analysis of deep networks”. In: *International conference on machine learning*. PMLR. 2021, pp. 4287–4296.

[15] T. Viering and M. Loog. “The shape of learning curves: a review”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.6 (2022), pp. 7799–7819.

[16] R. Sutton. “The bitter lesson”. In: *Incomplete Ideas (blog)* 13.1 (2019), p. 38.

[17] P. Villalobos, A. Ho, J. Sevilla, T. Besiroglu, L. Heim, and M. Hobbhahn. “Will we run out of data? Limits of LLM scaling based on human-generated data”. In: *arXiv preprint arXiv:2211.04325* (2022).

[18] X. Wang, I. Alabdulmohsin, D. Salz, Z. Li, K. Rong, and X. Zhai. “Scaling Pretraining to One Hundred Billion Data for Vision Language Models”. In: *arXiv preprint arXiv:2502.07617* (2025).

[19] A. Fu, J. Tran, A. Xie, J. Spraggett, E. Ding, C.-W. Lee, K. Singla, M. S. Hosseini, and K. N. Plataniotis. “NoFADE: Analyzing Diminishing Returns on CO2 Investment”. In: *arXiv preprint arXiv:2111.14059* (2021).

Acknowledgments

Jan, I want to thank you for being patience incarnate. Not just in matters of productivity, but also in the many times I would not take the time to reflect deep enough on your words and rather react directly to the feedback gleaned at the surface level. Please keep giving those after me the space and freedom I experienced in my research. You taught me to look for who "why?" of it all, to be pragmatic outside of research, but most importantly to sometimes just shut up, and listen. I hope to carry these values forward in my career.

Marcel, thank you for cultivating such a wonderful group. I also sincerely appreciate the pushback you gave me on my bold opinions on collaboration and the broadness of the group. You made me see a wider perspective than my own, even through my stubbornness.

Then, to my colleagues, office roommates and most importantly, friends:

Attila, you embody a wonderful combination of laissez-faire and control. I learned early on I could trust you with all kinds of practical matters. I am proud of the workshop and paper we did together. Most of all I enjoyed your company: always in for some banter, throwing a ball around (sorry Ombretta) or some feedback on potential names for the hot new architecture. I cherish our friendship, to be continued hopefully in more (ski) trips.

Ombretta, I could not wish for a nicer friend than you. You are so kind, even though I treated you to a way too direct joke about your competition for the PhD spot you beat me in, on your first day. I think we were both happy to find a colleague to confide in about the ups and downs of our job. We helped each other through rejections and COVID, and came out of it all with a strong friendship. I hope Fleur and I will see you and **Jonathan** still for the skip trips, Eurovision nights and to again share the ups and downs of our lives.

To those who welcomed me into the CV lab: **Osman**, a big smile with a big heart. Thank you for showing us it's okay to take breaks. Your bright-mindedness set the tone for my PhD experience. **Nergis**, I admire how you combine work with family life. I am happy for you that you were able to develop both, since we got to know each other as office roommates. Thank you for your outside perspective on CV, from which I learned much. **Silvia**, your assured and confident posture in the lab early on during my tenure set an important feminist example for me. I was always glad to see you around after you left the lab. You

are a warm presence to have around. **Yancong**, I admire your perseverance, whether it is in learning to ski together or your PhD work. Your generous laugh can seemingly come out of nowhere. I hope to say in touch and see you on the beginner slopes together once again. **Seyran**, Architecture should be glad to have you. I've enjoyed our meetings and trips like BMVC, where we could share our sense of humor. **Xin**, thank you for your friendliness, from when we first sat next to each other on the fifth floor.

To those who would gladly join in the Swedish *fika* coffee breaks: **Rickard**, you are incredibly kind and attentive. I have no doubts you will make it far in whatever career you choose. Thank you for introducing me to *fika* and calling me out on bullshit when necessary. Also, I'm sorry I made you talk Swedish to my Swedish-obsessed father at my wedding for more than half an hour. **Aurora**, we've had a lot of fun in the short time so far. I have no doubt your spiking nets will be successful, and I hope you and Ombretta will keep partying! **Hesam**, you are a cheerful and friendly presence in the group, which is much appreciated. **Alejandro**, I am glad to see you've found your place in Delft, though I still hope you can get used to our weather. **Sander**, I am still jealous of your PhD topic. We've had a lot of fun, and I trust you'll so well in your research. We even already have a paper together! **Casper**, I love how excited you are about life and research. Keep that! **Xucong**, you have such an expressive and happy face. I hope for you that you can find a way to spend more time with your wife. **Xiangwei**, you've showed yourself to me as ranging from diligent and focused in work to possessing a goodhearted laugh. **Burak**, I admire the strong perseverance you have shown in your time with us on the fifth floor, but also in the dedication to visit Delft for a work day all the way from Groningen. **Marian**, I don't think I know a more enthusiastic person than you. It is a contagious attitude! **Hadi**, though we only briefly worked together I can see your know-how and industry attitude are a valuable addition to the lab. Many thanks also go out to **Yeshwant**, **Yunwiang**, **Amongh**, **Nikolaas** and **Ziqi**, for good times and valuable feedback.

The Pattern Recognition lab has had a large impact on my PhD experience and my current academic attitude, for which I am immensely thankful. **David**, you inspire us all to put curiosity first in academia. I aspire to have the same level of enthusiasm and humor in my own talks and lectures as you do. Thank you for setting a great example to impressionable PhD's like me. **Marco**, it would be too easy to only mention how much I appreciate your humor. Though you have so much experience, you keep an open mind to academic as well as inter-social matters. Also, thanks for teaching me (by way of ontbijtkoek) that rules don't solve everything. **Tom**, I've seen first hand how you changed from a very vocal and assertive group presence to someone open to admitting they don't know something, and I recognize this as an example for my own attitude. Thank you

for your generous laughter to cheer us op. **Jesse**, from your first day as a PI in our group it was clear to me how much expertise you bring. I've also gotten know you as a friendly and reliable voice in the group. **Arman**, I enjoy our banter in the hallways of the sixth floor. I hope we continue this when I sporadically visit Delft. **Taylan**, I admire your strong value to represent those less vocal in the group. Please keep being the colorful representative you are. **Mahdi**, you appear calm and quiet, but I've seen you find your place in Delft and open up. I hope to see your smile even more often in the future. **Ramin**, you seemingly cannot pass me in the hallway without giving a wink. I love how expressive you are. To **Merwe**, **Yuko**, **Skander**, **Stephan** and **Jin**, I wish you the very best and thank you for the company.

To the affectionately called "Social bla bla" lab: **Hayley**, thank you for your sage advice about doing something different in my career. I see now the value in this. Thanks for all your wise contributions to the group and to me. **Chirag**, I was so happy to have a conspirator in initiative: from the self-appointed task to tour all new group members, to our collaboration on collaboration. Please keep this spark, no matter how much academia may stay in your way. And of course, many thanks also to **Bernd**, **Stephanie**, **Jose** and **Tiffany** for all our interactions.

To those from "Bio" I had the pleasure of knowing: **Thomas**, thank you for being direct, for organizing and taking leadership on cluster matters and for having a laugh in the hallways. **Aysun**, I appreciate how you used your voice for the benefit of others, for example in your DITTO representative role, but also in our collaboration on collaboration. **Swier**, I am glad we got to know each other on the ski trip. Your fearlessness and relaxed attitude are an inspiration. Thanks also to the many faces that I gained familiarity with through the shared social room: **Stavros**, **Tamim**, **Amelia**, **Kirti**, **Yasin**, **Paul**, **Sander**, **Mostafa**, **Ramin**, **Lieke**, **Colm**, **Jana**, **Jasmijn** and **Joana**. I will miss your trusted Thursday company.

Then there are a few "outside" people without whom this thesis would not have been possible. **David Romero**, thank you for letting me into your CKConv world. I learned a lot from our collaboration. I'm glad you found the opportunity to keep working on this and many other interesting topics. To **prof. Matthias Bethge** in Tübingen, thank you so much for letting me enjoy the summer-long visit of your lab. To **Matthias Kümmerer**, **Thaddäus** and the many others there: thank you for filling my time there with so many happy memories that I chose to come back as a postdoc. I'm sorry I could not make it in the end.

To the students I was lucky to supervise: **Leonid**, **Tomasz**, **Xiangxie**, **Pierre**, **Mark** and **Luuk**, I learned a lot from our interactions, and something different from each of you. Thank you for your work on some of the chapters in this book. It would not have been possible without you.

I would be remiss to forgot to mention the incredible technical and administrative support from **Ruud**, **Bart**, **Azza**, **Marunka** and **Saskia**. Thank you for allowing us to focus on our research, day in, day out.

To **Thomas Mensink**, my early mentor as a student at the University of Amsterdam, who put me on to path to apply with Jan: I am immensely thankful for your generous and wise counsel.

Finally, to my **friends and family** close to me, who have supported me personally. Even when finally the combined stress of graduating, marrying and everything else burned me out, you got me through it. **Pap en mam**, ik hou van jullie. Dank jullie wel voor al jullie steun. En als laatste, mijn allerliefste **Fleur**: jij weet net zo goed als ik dat ik dit alles, tot en met de burnout aan toe, niet zonder jou had kunnen doen. Jij inspireert me elke dag om een beter mens te zijn, ook in mijn werk. Ik hou van jou.

Robert-Jan Bruintjes
Utrecht, June 2025

List of Publications

In this thesis

1. R. Bruintjes, A. Lengyel, O. S. Kayhan, D. Zambrano, N. Tömen, H. Jamali-Rad, and J. van Gemert. "Data-Efficient Challenges in Visual Inductive Priors: A Retrospective". In: *arXiv preprint arXiv:2506.08612* (2025)

2. R. Bruintjes, T. Motyka, and J. van Gemert. "What Affects Learned Equivariance in Deep Image Recognition Models?" In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2023, pp. 4839–4847

3. R. Bruintjes and J. van Gemert. "Learning to Adapt to Position Bias in Vision Transformer Classifiers". In: *arXiv preprint arXiv:2505.13137* (2025)

4. M. Basting, R. Bruintjes, T. Wiedemer, M. Kümmerer, M. Bethge, and J. van Gemert. "Scale Learning in Scale-Equivariant Convolutional Networks". In: *VISIGRAPP (2): VISAPP*. 2024

5. D. W. Romero, R. Bruintjes, J. M. Tomczak, E. J. Bekkers, M. Hoogendoorn, and J. van Gemert. "FlexConv: Continuous Kernel Convolutions With Differentiable Kernel Sizes". In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: https://openreview.net/forum?id=3jooF27-0Wy

Other publications

1. R. Bruintjes, A. Lengyel, M. Baptista-Ríos, O. S. Kayhan, D. Zambrano, N. Tömen, and J. van Gemert. "VIPriors 4: Visual Inductive Priors for Data-Efficient Deep Learning Challenges". In: *arXiv preprint arXiv:2406.18176* (2024)

2. A. Lengyel, O. Strafforello, R. Bruintjes, A. Gielisse, and J. van Gemert. "Color Equivariant Convolutional Networks". In: *Advances in Neural Information Processing Systems*. Vol. 36. 2023, pp. 29831–29850

3. J. Warchocki, T. Oprescu, Y. Wang, A. Dămăcuş, P. Misterka, R. Bruintjes, A. Lengyel, O. Strafforello, and J. van Gemert. "Benchmarking Data Efficiency and Computational Efficiency of Temporal Action Localization Models". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*. Oct. 2023, pp. 3008–3016

4. R. Bruintjes, A. Lengyel, M. Baptista-Ríos, O. S. Kayhan, D. Zambrano, N. Tömen, and J. van Gemert. "Vipriors 3: Visual inductive priors for data-efficient deep learning challenges". In: *arXiv preprint arXiv:2305.19688* (2023)

5. A. Lengyel, R. Bruintjes, M. Baptista-Ríos, O. S. Kayhan, D. Zambrano, N. Tömen, and J. van Gemert. "Vipriors 2: visual inductive priors for data-efficient deep learning challenges". In: *arXiv preprint arXiv:2201.08625* (2022)

6. R. Bruintjes, A. Lengyel, M. Baptista-Ríos, O. S. Kayhan, and J. van Gemert. "VIPriors 1: Visual Inductive Priors for Data-Efficient Deep Learning Challenges". In: *arXiv preprint arXiv:2103.03768* (2021)

7. T. Das, R. Bruintjes, A. Lengyel, J. van Gemert, and S. Beery. "Domain adaptation for rare classes augmented with synthetic samples". In: *arXiv preprint arXiv:2110.12216* (2021)

8. A. Schuth, R. Bruintjes, F. Buüttner, J. van Doorn, C. Groenland, H. Oosterhuis, C.-N. Tran, B. Veeling, J. van der Velde, R. Wechsler, *et al.* "Probabilistic multileave for online retrieval evaluation". In: *Proceedings of the 38th international ACM SIGIR Conference on Research and Development in Information Retrieval*. 2015, pp. 955–958

Curriculum Vitæ

Robert-Jan Bruintjes

23 May 1992 Born in Amersfoort, the Netherlands.

Education

2019–2026 **PhD in Computer Science**
Delft University of Technology, Delft
Thesis: Learning with Spatial Biases in Vision Models
Promotors: Dr. J.C. van Gemert
Prof. dr. ir. M.J.T. Reinders

2014–2017 **Master of Science in Artificial Intelligence**
University of Amsterdam, Amsterdam

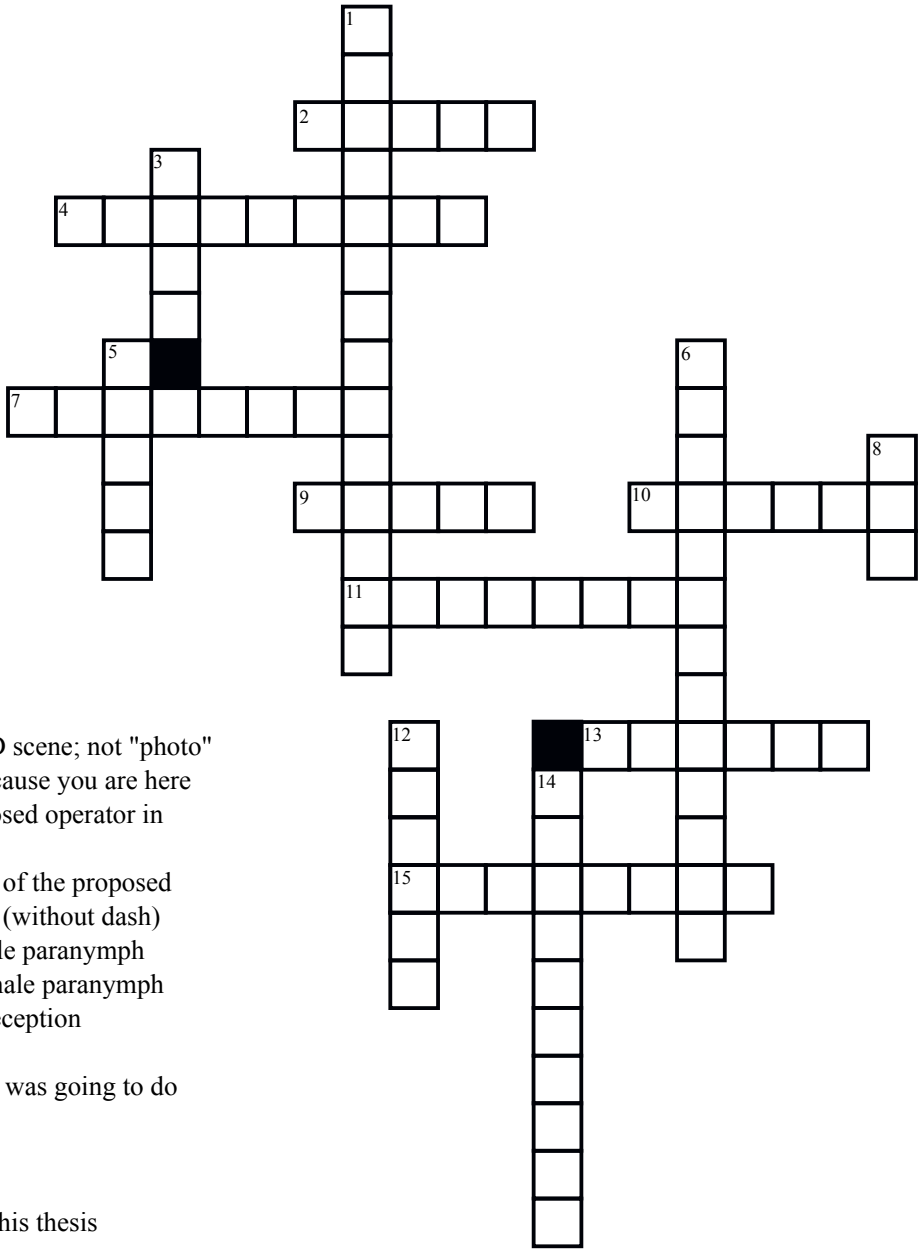
2010–2014 **Bachelor of Science in Computer Science**
Utrecht University, Utrecht

2004–2010 **Pre-university education (VWO)**
Greijdanus College, Zwolle

Work Experience

2025 - current **Autoriteit Persoonsgegevens, The Hague**
Senior Technical Advisor

2017–2019 **Sightcorp, Amsterdam**
Lead Deep Learning Engineer



Across

2. 2D version of a 3D scene; not "photo"
4. Feeling RJ has because you are here
7. Name of the proposed operator in chapter 6
9. Abbreviated name of the proposed measure in chapter 4 (without dash)
10. First name of male paranymph
11. First name of female paranymph
13. Free ___ at the reception afterwards!
15. German place RJ was going to do a postdoc

Down

1. Research field of this thesis (two words)
3. Needed to train a vision model
5. Number of years it took RJ to graduate (written out)
6. Learning with ___ in Vision Models (two words)
8. First name of promotor
12. Title obtained after successful graduation
14. Number of red balls (not counting flat circles) in/on this thesis

Answers are given on the next page

Crossword answer key

Across

2. image
4. gratitude
7. FlexConv
9. PSHAP
10. Attila
11. Ombretta
13. drinks
15. Tübingen

Down

1. Computer Vision
3. data
5. seven
6. Spatial Biases
8. Jan
12. doctor
14. thirtynine

The field of computer vision research is very large and still growing. Many of these papers concern some type of inductive bias, by proposing new building blocks or alternative training methods for vision models. This type of research has enabled great progress in applications of vision models.

Computer Vision concerns itself with research and development of Deep Learning models that work on visual data. These vision models are already integrated heavily in society, powering applications in the real world, such as automated radiology in the hospital, self-driving cars and autonomous drones. However, it takes a lot of data, in the form of datasets of thousands or millions of images, to learn reliable vision models. This thesis explores which role spatial biases (prior knowledge on the position and pose of objects in the image) can play in learning better and more data-efficient vision models.

We find that the practice of integrating prior knowledge on spatial biases (inductive spatial biases) can help to learn biases that are otherwise hard or impossible to learn. Though inductive bias can be difficult and time-consuming to design, and often increases inference cost, integrating inductive bias can result in better performance and better data-efficiency. This work showcases these patterns in spatial biases, specifically position bias and scale bias.

We find position bias may be learned to some degree by models without the proper inductive bias, but inductive bias helps to model these biases and improves performance. We show it depends on the data if learning position bias is helpful for models. We contribute measures for position bias in vision models in general, as well as in Vision Transformers specifically, to enable discovering these findings. We propose an inductive bias on the position embedding of ViTs to better (un)learn position bias.

For scale bias, we find that existing scale equivariant models for scale bias need to be tuned to the scale distribution of the data. We propose an inductive bias that allows scale equivariant models to learn the scale bias of the dataset, therefore fitting better to the data. We also propose an alternative parameterization of convolutions called MAGNet that can be adapted to known scale distributions present in the data. Models using MAGNets (FlexNets) can be much shallower and do not require pooling.

There are those who advocate against spending much time on inductive biases. The "bitter lesson" of Richard Sutton prescribes that we should just add more data, not more inductive bias. However, data will run out at some point, perhaps sooner rather than later. Besides, raw performance of vision models, given as much data as possible, should not be our only goal: data-deficient settings are real, plenty and important. Data-efficient vision models are the future of our field, and the search for appropriate inductive biases will remain an important endeavor.