



Generalisation in deep convolutional neural networks

NR Molapo



orcid.org/ 0000-0002-0112-9098

Thesis accepted in fulfilment of the requirements for the degree
Doctor of Philosophy in Computer and Electronic Engineering at the
North-West University

Promoter: Prof E Barnard

Graduation: June 2025

Declaration

I, Nkadimeng Raymond Molapo, hereby declare that the dissertation entitled “Generalisation in deep convolutional neural networks” is my own original work and has not already been submitted to any other university or institution for examination.



N.R. Molapo

Student number: 23380845

Signed on the 14th day of April 2025 at Pretoria, Gauteng, South Africa.

Acknowledgements

First and foremost, I would like to express my sincerest gratitude to Prof. Etienne Barnard, my supervisor, for his incredible guidance, honest appraisal, patience and willingness to share his wealth of knowledge throughout this journey.

I am grateful to the MUST Deep Learning team for their encouragement and for creating a community of support. A special thanks to Prof. Marelie Davel for her leadership and ensuring I focused and saw the bigger picture.

To Ulrike Janke, I am indebted to your kindness, support and ensuring that all administrative processes are taken care of.

Finally, thank you to my friends for your understanding and enduring support during this exciting journey. To my immediate family, this would not have been possible without your sacrifices and unwavering support.

Abstract

Understanding the remarkable generalization abilities of Deep Learning (DL) systems remains one of the significant scientific challenges of our time. Even though it is widely accepted that the success of Deep Neural Networks stems (at least partially) from having many hidden layers. However, the benefits of such depth are not universal. For example, in fully connected networks or Multilayer Perceptrons (MLPs), it seems likely that networks with a single hidden layer can be found to generalize as well as deeper networks. In contrast, Convolutional Neural Networks (CNNs) demonstrate a relatively robust trend of improved generalization with greater network depth.

We introduce a simple experimental paradigm demonstrating the contrast between CNNs and MLPs on the MNIST, FMNIST and CIFAR-10 datasets. This paradigm demonstrates a critical distinction between modern networks and their predecessors, which is the ability of contemporary architectures to leverage deep, multi-layered structures. It consistently demonstrates improved generalization as the number of hidden layers increases. These observations, however, conflict with the classical learning theory and its key concept of the bias-variance tradeoff, which implies that over-parameterized, high-dimensional and high-complexity networks should not perform well on unseen data.

Therefore, we present an alternative framework for understanding the relationship between network architecture and generalization by viewing classifiers as maps between different metric spaces. This explicit expression, represented by MLPs and CNNs with Rectified Linear Unit (ReLU) transfer functions, is derived to show that maps for CNNs are biased to be increasingly smooth with greater depth. Since such smoothness is a characteristic of real images, deeper CNNs tend to outperform shallow CNNs and MLPs.

Through comparative analysis, we uncover how deeper networks develop a bias towards smoother input representations. We further argue that Wolpert’s Extended Bayesian Framework (EBF) offers a promising theoretical lens to characterise this bias. While fully harnessing the insights provided by the Extended Bayesian Framework (EBF) will necessitate the creation of new theoretical and computational tools, this approach can

potentially deepen our understanding of generalization in Deep Learning.

Keywords: *Generalization, Convolutional Neural Networks, Deep Neural Networks, Deep Learning, Interpolation*

Contents

List of Figures	9
List of Tables	14
List of Acronyms	16
1 Introduction	1
1.1 Evolution of deep neural networks	1
1.2 Problem statement and research question	3
1.3 Research contributions	4
1.4 Thesis overview	4
2 Literature review	6
2.1 Overview	6
2.2 Bias-variance trade-off	7
2.2.1 Dominant dogma	7
2.2.2 Overparameterized networks	9
2.2.3 Double descent	9
2.3 Sharpness in the weight landscape	11
2.3.1 Simplified flatness	11
2.3.2 A deeper look at flatness	12

2.3.3	Sharpness aware minimization	14
2.4	Margins	15
2.4.1	Linear considerations	15
2.4.2	Non-linear considerations	16
2.5	Expressivity and learnability	17
2.6	Implicit generalization and effective complexity	18
2.7	Geometric deep learning and interpolation	19
2.7.1	Interpolation variations	20
2.8	Discussion	20
3	Benefits of depth in CNNs	23
3.1	Overview	23
3.2	Experimental setup and generalization	24
3.3	Data	25
3.4	Results	27
3.5	Discussion	32
4	Piecewise linear model	38
4.1	Overview	38
4.2	Classifiers as converters	39
4.3	Input-output mapping in a CNN	41
4.4	Linear representation of a convolution layer	43
4.5	Interpolation between training samples	44
4.6	Results	46
4.6.1	Binary image interpolation	46
4.6.2	2D FFT binary images	53

4.6.3	Bit change summations	55
4.7	Discussion	65
5	Conclusion	67
5.1	Overview	67
5.2	Key findings	68
5.2.1	Investigating importance of depth in CNNs	68
5.2.2	Investigating the piecewise linear model	69
5.3	Future work	70
5.4	Final remarks	71
	References	73
A	Supplemental result plots	83
A.1	Appendix: Chapter 3	83
A.2	Appendix: Chapter 4	86

List of Figures

2.1	Bias-variance tradeoff [17] plot shows low complexity model with high bias turns to underfit the data while a high-complexity model with high variance tends to overfit. However, a zone exists between the dotted lines where the model will likely generalize well.	8
2.2	A plot of the test loss as the sum of the bell-shaped variance and monotonic bias. Case a) where the bias is much larger than the variance, resulting in a monotonically decreasing risk curve. Case b) where the variance and bias alternate in regime dominance, resulting in a double descent risk curve. Case c) where the variance is much larger than the bias, resulting in a bell-shaped risk curve that decreases after the threshold. Adapted from [25].	12
2.3	A sketch of valleys with flat minima vs. sharp minima. This illustrates the high sensitivity of a training function to perturbations, which negatively affects generalization. Regions with flat minima are less complex and require low precision weights, making them likely to generalize better. Adapted from [33].	13
2.4	a) A sketch of the maximum margin classifier shows that positive and negative hyperplanes can separate the two classes but not with maximum margin. The maximum margin hyperplane maximises the separation from its closest point from each class. b) To compute the hypothesis margin at x for a set of points P , $\text{nearmiss}(x)$ and $\text{nearhit}(x)$ denote the closest point to x in P with different and same labels, respectively.	18
3.1	An architectural design of a lower capacity deep CNN with average pooling and a single 20-neuron fully connected layer. The network depth of 4 layers was only constrained by the input dimensions. Note that the dimensionalities shown are for FMNIST and MNIST datasets.	25

3.2	An architectural design of a lower capacity deep CNN with average pooling and a single 20-neuron fully connected layer. The network depth of 4 layers was only constrained by the input dimensions. Note that the dimensionalities shown are for the CIFAR-10 dataset.	26
3.3	An architectural design of an MLP network. Lower capacity networks utilised 512 neurons, while higher capacity networks utilised 1024 neurons.	26
3.4	a) MNIST and b) Fashion MNIST single-channel class dataset samples. Class members are grouped vertically for all 10 classes.	28
3.5	CIFAR-10 3-channel class dataset samples. Class members are grouped vertically for all 10 classes.	28
3.6	Validation error rate and loss per CNN network depth for MNIST dataset. a) represents a lower capacity network that starts from 32 neurons in the first layer and doubles in subsequent layers. b) represent a higher capacity network that starts from 64 neurons and doubles in subsequent layers. . . .	29
3.7	Validation error and loss per MLP network capacity for MNIST, FMNIST and CIFAR-10 datasets. 1024 neurons represent higher capacity, while 512 neurons represent lower capacity models.	30
3.8	Average validation error rate and validation loss per CNN network depth for FMNIST dataset. a) represents a lower capacity network that starts from 32 neurons in the first layer and doubles in subsequent layers. b) represent a higher capacity network that starts from 64 neurons and doubles in subsequent layers. Note that 5 models were averaged to generate the plots.	32
3.9	Average validation error rate and loss per CNN network depth for CIFAR-10 dataset. a) represents a lower capacity network that starts from 32 neurons in the first layer and doubles in subsequent layers. b) represent a higher capacity network that starts from 64 neurons and doubles in subsequent layers. Note that 5 models were averaged to generate the plots. . . .	33
4.1	Image (c) is shifted one pixel to the right compared to image (a); hence, the respective Euclidean distances between images (b) and (c) to image (a) are the same.	41
4.2	MNIST standard and binary samples. Samples on the left represent binary images with only 0s and 1s. Note that all pixel values greater than 0 were assigned to 1.	47
4.3	MNIST class 0 intra-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.	48

4.4	MNIST class 1 intra-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.	48
4.5	MNIST class 2 intra-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.	49
4.6	MNIST class 3 intra-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.	49
4.7	MNIST class 4 intra-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.	50
4.8	MNIST class 5 intra-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.	50
4.9	MNIST class 6 intra-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.	51
4.10	MNIST class 7 intra-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.	51
4.11	MNIST class 8 intra-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.	52
4.12	MNIST class 9 intra-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.	52
4.13	MNIST classes 4 and 8 inter-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.	53
4.14	MNIST class 2 sample interpolation with 2D Fast Fourier Transform (FFT) for the different CNN depths and an MLP network. Each subsequent interpolation stage was transformed from Figure 4.5.	55

4.15	An intermediate binary shifted 2D FFT plot for a) single symmetric interpolated image for a class 2 MNIST sample. b) clear symmetric illustration of the even and odd quadrants with centred frequencies. c) half 2D image sample for frequency calculations without symmetry redundancy. d) half 2D sample with $\mathbf{R} = \mathbf{4}$ mesh grid for frequency summation.	56
4.16	MNIST intra-class average 2D FFT sums for 4 CNNs and MLP network. The numbers within the legend represent the depth of each CNN and the MLP represents a single-layer, large-capacity network. Each interpolation point is computed using an average of 10 2D FFT sums.	57
4.17	FMNIST intra-class average 2D FFT sums for 4 CNNs and MLP network. The numbers within the legend represent the depth of each CNN and the MLP represents a single-layer, large-capacity network. Each interpolation point is computed by averaging over 10 2D FFT sums.	58
4.18	MNIST inter-class average 2D FFT sums for 4 CNNs and MLP network. The numbers within the legend represent the depth of each CNN and the MLP represents a single-layer, large-capacity network. Subplot titles indicate the different interpolated classes. Each interpolation point is computed by averaging over 10 2D FFT sums.	59
4.19	MNIST inter-class bit count for 4 CNNs and MLP network. The numbers within the legend represent the depth of each CNN and the MLP represents a single-layer, large-capacity network. Subplot titles indicate the different interpolated classes. Each interpolation point is computed by adding vertical and horizontal bit changes.	60
4.20	MNIST intra-class bit counts for 4 CNNs and MLP network. The numbers within the legend represent the depth of each CNN and the MLP represents a single-layer, large-capacity network. Each interpolation point is computed by adding vertical and horizontal bit changes.	62
4.21	FMNIST intra-class bit counts for 4 CNNs and MLP network. The numbers within the legend represent the depth of each CNN and the MLP represents a single-layer, large-capacity network. Each interpolation point is computed by adding vertical and horizontal bit changes.	63
4.22	FMNIST inter-class bit count for 4 CNNs and MLP network. The numbers within the legend represent the depth of each CNN and the MLP represents a single-layer, large-capacity network. Subplot titles indicate the different interpolated classes. Each interpolation point is computed by adding vertical and horizontal bit changes.	64

A.1	a) MNIST and b) Fashion MNIST dataset class membership for training and test sets. All class members in the Fashion MNIST training and test sets are balanced while memberships in MNIST show a slight imbalance. .	84
A.2	CIFAR 10 dataset class membership for training and test sets. All class members in the training and test sets are balanced.	84
A.3	Architectural design of a higher-capacity deep CNN with average pooling and a single 20-neuron fully connected layer. The network depth of 4 layers was only constrained by the input dimensions. Note that the dimensionalities shown are for FMNIST and MNIST datasets.	85
A.4	architectural design of a higher-capacity deep CNN with average pooling and a single 20-neuron fully connected layer. The network depth of 4 layers was only constrained by the input dimensions. Note that the dimensionalities shown are for CIFAR-10 dataset.	85
A.5	MNIST classes 4 and 8 inter-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.	86
A.6	MNIST classes 4 and 8 inter-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.	86

List of Tables

3.1	Average lower capacity MLP (represented by 0 CNN layers) and CNN performance results for increasing depth on the MNIST validation sets. Each number in the filters column represents the number of neurons per subsequent convolutional layer. Note that the number of neurons in the fully connected layer in CNNs was set to 20 for all experiments. Validation error was averaged over 5 trained models and standard error represents how spread out the values are around the mean.	31
3.2	Average higher capacity MLP (represented by 0 CNN layers) and CNN performance results for increasing depth on the MNIST validation sets. Each number in the filters column represents the number of neurons per subsequent convolutional layer. Note that the number of neurons in the fully connected layer in CNNs was set to 20 for all experiments. Validation error was averaged over 5 trained models and standard error represents how spread out the values are around the mean.	34
3.3	Average lower capacity MLP (represented by 0 CNN layers) and CNN performance results for increasing depth on the FMNIST validation sets. Each number in the filters column represents the number of neurons per subsequent convolutional layer. Note that the number of neurons in the fully connected layer in CNNs was set to 20 for all experiments. Validation error was averaged over 5 trained models and standard error represents how spread out the values are around the mean.	35
3.4	Average higher capacity MLP (represented by 0 CNN layers) and CNN performance results for increasing depth on the FMNIST validation sets. Each number in the filters column represents the number of neurons per subsequent convolutional layer. Note that the number of neurons in the fully connected layer in CNNs was set to 20 for all experiments. Validation error was averaged over 5 trained models and standard error represents how spread out the values are around the mean.	35

3.5	Average lower capacity MLP (represented by 0 CNN layers) and CNN performance results for increasing depth on the CIFAR-10 validation sets. Each number in the filters column represents the number of neurons per subsequent convolutional layer. Note that the number of neurons in the fully connected layer in CNNs was set to 20 for all experiments. Validation error was averaged over 5 trained models and standard error represents how spread out the values are around the mean.	36
3.6	Average higher capacity MLP (represented by 0 CNN layers) and CNN performance results for increasing depth on the CIFAR-10 validation sets. Each number in the filters column represents the number of neurons per subsequent convolutional layer. Note that the number of neurons in the fully connected layer in CNNs was set to 20 for all experiments. Validation error was averaged over 5 trained models and standard error represents how spread out the values are around the mean.	36
3.7	Average MLP performance results for increasing depth on the MNIST, FMNIST and CIFAR-10 validation sets. Average validation error (Avg) and standard error (Std) thereof computed over 5 trained models starting from different random initializations.	37

List of Acronyms

ANNs Artificial Neural Networks

CNNs Convolutional Neural Networks

DL Deep Learning

DNNs Deep Neural Networks

EBF Extended Bayesian Framework

FFT Fast Fourier Transform

GDL Geometric Deep Learning

GPUs Graphical Processing Units

kNN k-Nearest Neighbor

LLMs Large Language Models

ML Machine Learning

MLPs Multilayer Perceptrons

PAC Probably Approximately Correct

ReLU Rectified Linear Unit

SAM Sharpness Aware Minimization

SGD Stochastic Gradient Descent

SGLD Stochastic Gradient Langevin Dynamics

SLT Statistical Learning Theory

SVM Support Vector Machine

Chapter 1

Introduction

In this chapter, we discuss the evolution of deep neural networks and briefly introduce the outstanding issue that motivates our problem statement. We also summarize our research contributions and give an overview of the thesis.

1.1 Evolution of deep neural networks

The current wave of excitement surrounding Large Language Models (LLMs) is based on the fifth generation of Artificial Neural Networks (ANNs), which – as with each of the preceding generations – represents a speculative combination of neurobiological, mathematical, and engineering insights. It is helpful to briefly review the history of these architectures to understand the current research goals.

McCulloch and Pitts described the earliest ANNs in 1943 [1]. These were no more than networks of simple logic gates and could not learn. Thus, designing networks that could perform valuable tasks was a challenge that quickly became insurmountable as task complexity increased.

An algorithm for learning in such networks was introduced by Rosenblatt in 1958. His “perceptrons” attracted significant attention, but early expectations of their capabilities did not materialize. When Minsky and Papert [2] proved certain fundamental limitations of perceptron networks, enthusiasm for ANNs waned.

The third generation of ANNs arose in the mid-1980s. Several research groups then suggested innovations that could resolve the concerns of Minsky and Papert or expand the capabilities of ANNs in other ways. Most significantly, the “backpropagation” algorithm [3] was able to train so-called Multilayer Perceptrons (MLPs), which directly addresses the most substantial concern of [2]. As with the previous generations, a wave of excitement dissipated when initial successes did not extend to match the visions that early developments had suggested. Performance tended to plateau at unsatisfactory levels for quantities of training data that were manageable by state-of-the-art algorithms and processors.

These limitations were overcome dramatically with the Deep Neural Networks (DNNs) developed by Hinton and collaborators after 2007 (see, for example, [4]). Utilizing Graphical Processing Units (GPUs) to accelerate training and several algorithmic enhancements, these networks seamlessly rose above the performance ceilings previously observed. The most influential networks in this generation were Convolutional Neural Networks (CNNs), and they were “deep” in the sense that networks with multiple layers of nodes were found to outperform “shallower” networks. This was in sharp contrast to MLPs, where even one layer of nodes intermediate to the inputs and outputs was found to be optimal (or close to optimal) [5].

The latest generation of ANNs is known as transformer networks [6]. These networks were developed to address the sequence-to-sequence mapping task that arises in applications such as machine translation. transformers proved extremely powerful in this regard, and researchers quickly found ways to cast a wide range of applications as sequence-to-sequence tasks: language modelling [7], [8], speech recognition [9], [10], computer vision [11], [12], and even robotics.

Whereas the earlier generations had all appeared in responses to observed limitations of their predecessors, transformers were developed even while CNNs were being extended with great success to new domains. Consequently, much of the theoretical focus in ANNs has switched to transformer architectures even though fundamental shortcomings with CNNs were not well understood. Most obviously, the benefit of multiple hidden layers that occur in CNNs but not MLPs received substantial attention in the years before the development of transformers without satisfactory resolution. The literature contains interesting hints [13] that this phenomenon conflicts with some long-held beliefs in Computational Learning Theory by revealing that overparametrized networks may perform optimally (which would not be predicted from the classical bias-variance trade-off [14]).

Given that CNNs are still practically important (especially in applications such as computer vision) and that insights here may also help understand transformers and future architectures, we have chosen to investigate this issue in the current research. In particular, we present additional evidence of the contrast between CNNs and MLPs with respect to the number of computational layers and then study the root causes of this contrast.

1.2 Problem statement and research question

It is widely accepted that extended depth in overparameterized CNNs is instrumental in yielding excellent generalization performance [13], [15], [16]. Even so, the benefits of such depth are not universal. For example, single-layer feedforward MLP networks of adequate capacity can be found to provide similar generalization performances to any deep MLP network (Walter Heymans, private communication). In addition, the theoretical reasoning behind the findings for Rectified Linear Unit (ReLU) CNNs seem to contradict the enduring credence in Statistical Learning Theory (SLT), which has not been adequately addressed and is still an open area of research. Our main research question is, therefore, as follows: why do deep CNNs generalize so well, and what does that teach us about the general mechanisms of generalization?.

1.3 Research contributions

The main contributions of this work with regard to existing knowledge include:

- A systematic demonstration that significant generalization differences for certain types of networks do not follow the classical concept of a bias-variance trade-off described by SLT. Although much has been accomplished, a simple and unambiguous experimental framework is necessary to achieve the research goals. Interestingly, it is typically found that the loss in error due to bias-variance mismatch is in the order of a few per cent absolute. In contrast, we measure different depth networks with as much as 15-20 per cent absolute accuracy differences.
- A piecewise linear model is developed to perform efficient local calculations using CNNs.
- Evidence is provided that smoothing in deeper networks leads to large generalization differences. The a priori preference of the deeper networks is to provide improved smoothness in the neighbourhood of the data points, which is an appropriate bias to enhance generalization. Therefore, we can concretely demonstrate “inductive bias”, which, to date, has been a nebulous theoretical concept.

1.4 Thesis overview

This thesis consists of 5 chapters and is structured as follows:

- Chapter 2 reviews the relevant topics that influence the generalization ability in deep networks. These discussions provide further insights into similar works and set a scientific foundation for our research.
- Chapter 3 presents the empirical investigations of the effect of depth on the generalization ability of standard deep CNNs guided by the concepts introduced in the literature.

-
- Chapter 4 empirically explores our hypotheses on the benefits of network depth by observing the effects of localized linear mapping computed by standard deep CNNs.
 - Chapter 5 summarizes our findings and further expands on the most meaningful contributions of our research. It also provides guidelines for envisioned future work.

Chapter 2

Literature review

This chapter reviews the most prominent research into factors influencing the generalization ability in deep networks.

2.1 Overview

Generalization is the ability of a trained model to perform well on unseen data. In supervised Machine Learning (ML), labelled data is used to train an algorithm to classify data or predict a certain outcome. As training data is input into the model, it compares the output to the target through some loss function. For neural networks, the error of the loss function is propagated through the network to adjust the model parameters to minimize the error. This process is tailored to find the best solution for the data at hand (optimization) and unseen data (generalization). Since performance on unseen data is the central goal of pattern recognition, understanding its origin and characteristics is the main concern of theoretical research in ML.

This chapter briefly review the classical SLT perspective on the factors that influence the

generalization ability of deep neural networks and then summarise an example of more modern work that transcends SLT’s limitations. Section 2.2 introduces the concept of bias-variance trade-off and its relationship with model complexity. Section 2.3 summarises how the sharpness of the loss function can influence the generalization depending on its smoothness. Section 2.4 discusses research relating margin sizes to the prediction of generalization in deep neural networks. Section 2.5, discusses how structural properties of neural networks (in particular, network depth) can enable better function approximation without sacrificing generalization ability. Section 2.6 contains a brief summary of the various attempts to derive alternative definitions of model complexity from salvaging SLT and in Section 2.7, outlines Geometric Deep Learning (GDL), which shares some superficial similarities with our work.

2.2 Bias-variance trade-off

2.2.1 Dominant dogma

The basic concept of bias-variance tradeoff [14] refers to the compromise between a model’s ability to correctly capture the intrinsic data patterns and its responsivity to changes in the training data. The first factor, bias, is the average gap between the model’s prediction and the target value. In contrast, the second, variance, can be seen as the average variability of a model’s prediction for equivalent conditions (e.g., different data set samplings or training conditions). Complex ML models extract complex and convoluted patterns in the training data but can, in turn, model the noise, which results in high variance. Conversely, an oversimplified model representation is biased because it fails to capture the complexity of the data [17].

The bias-variance trade-off suggests that as the complexity of a model increases, its variance also increases but its bias decreases. Conversely, as the model complexity decreases, the variance decreases but the bias increases. This balance of complexities leads to a U-shaped test error curve depicted in Figure 2.1.

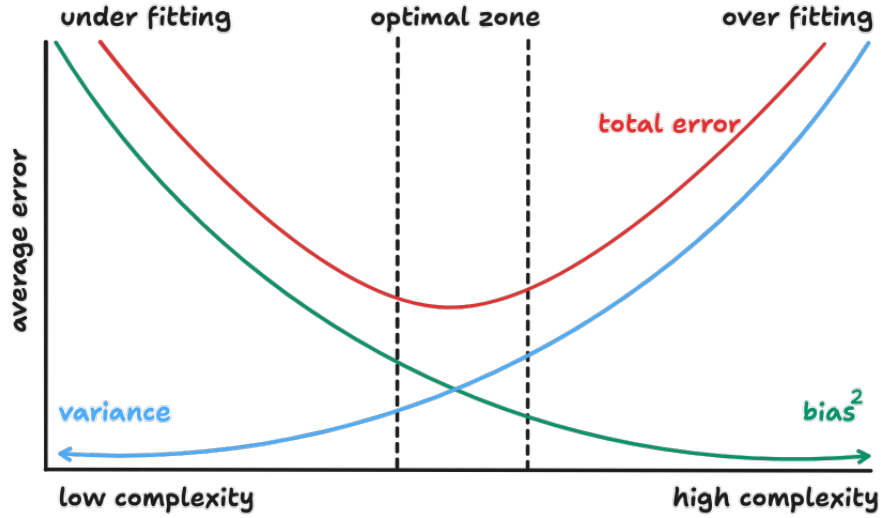


Figure 2.1: Bias-variance tradeoff [17] plot shows low complexity model with high bias turns to underfit the data while a high-complexity model with high variance tends to overfit. However, a zone exists between the dotted lines where the model will likely generalize well.

According to SLT, the ultimate aim is to strike a balance or compromise between the two extremes, shown between the dotted lines (optimal zone), which ensures that a model performs well on unseen data and generalizes well. As shown in Figure 2.1, a model that operates to the left of the optimal zone has high bias, is oversimplified, does not learn adequately from the training data and is said to underfit. In contrast, a model that operates to the right of the optimal zone has high variance and performs well on training data but poorly on unseen data and is said to overfit. Classical wisdom highlights that the trade-off is intrinsically associated with model complexity. In this frame of reference, finding a model with sufficient capacity to achieve the optimal bias-variance trade-off is necessary. Nonetheless, overparameterized architectures do not seem to suffer from the consequences of having excessive parameters as dictated by SLT. For this reason, classical norms and understandings may not be entirely suitable for contemporary deep learning networks.

2.2.2 Overparameterized networks

As mentioned in Section 2.2.1, classical wisdom emphasizes the trade-off associated with model complexity. In this frame of reference, finding a model with sufficient capacity to achieve the optimal bias-variance trade-off is necessary. Fortmann et al. [17] provided experimental evidence highlighting the bias-variance trade-off for non-parametric methods such as k-Nearest Neighbor (kNN) [18] and kernel regression [19] or in common classification and regression settings. The frame of reference is premised on increased network width being intrinsically linked to increased bias while the variance decreases. However, modern neural network architectures have frequently demonstrated the convenience of having many more parameters than classic wisdom dictates [12], [16]. This contradiction between modern practices and conventional wisdom suggests that some aspects of the classical theory may not be appropriate for understanding modern networks.

Recent works show that overparameterized networks do not exhibit the classic U-shaped error curve. This is attributed to the decrease in the test error for wider networks [20]–[22]. The absence of the U-shaped curve may imply the presence of some implicit regularization that regulates network capacity [23], as discussed in Section 2.6. However, a simple test error analysis does not reveal the reasons behind the absence of the bias-variance trade-off. A careful analysis needed to be conducted separately to investigate under which conditions both variance and bias increase or decrease. Research by Neal et al. [24] showed that the variance due to optimization increased significantly in the low-complexity spaces but decreased monotonically with width in high-complexity spaces.

2.2.3 Double descent

A deeper look into the analysis conducted by Neal et al. [24] reveals that while the bias decreases monotonically, the variance is unimodal or bell-shaped [25] (that is, it increases up to the interpolation threshold, then decreases in the overparameterized regime). Moreover, since the test loss is the sum of the bell-shaped variance and the monotonic bias, it can produce different outputs based on the magnitude of the bias and variance. In cases

where the bias is much larger than the variance, the outcome is a monotonically decreasing risk curve, as shown in Figure 2.2a. Conversely, in cases where the variance is much larger than the bias, the outcome is a bell-shaped curve that increases until the interpolation threshold and then decreases, as shown in Figure 2.2c. The last characteristic is when the variance and bias alternate in regime dominance, resulting in the double-descent risk curve shown in Figure 2.2b. This behaviour can be typically observed in extremely large models when labelled noise is incorporated into the training set.

As highlighted above, double descent is not seen in typical DNNs and relies heavily on various factors that may not be present in standard networks [26], [27]. In a recent publication, Schaeffer et al. [28] underline three interpretable factors preventing double descent from occurring. Firstly, the training features must have small to minuscule but non-zero variance in the same directions. Secondly, test features must be appreciably dissimilar to training features, and lastly, the best class model must not perfectly fit the training data. In contrast, Liu et al. [29] focused on factors contributing to the presence of double descent. These factors encompass batch size, normalization, initialization, learning rate and suitable optimization algorithm. As a result, Liu et al. assert that double descent cannot be seen if the chosen optimizer cannot settle on a flat and adequately low-loss minimum.

Belkin et al. [22] focused on the apparent contradiction between the bias-variance tradeoff and the practical observations of modern ML practices. Convention dictates that an optimal region should exist where a model can adequately learn the underlying data structure and be small enough not to fit noise patterns. The results should be a model that performs well on unseen data. A key observation is that effective model capacity is the main factor that regulates the trade-off. However, modern trends have shown that models are designed and trained to fit the training set. Conventionally, these models should overfit and not generalize well, but empirically, they do. Belkin et al. aim to reconcile modern practices and classical theory by incorporating varying bias-variance cases under a unified performance curve. Figure 2.2b comprises the U-shaped curve until the interpolation threshold indicated by the risk peak. The right side of the threshold can be observed when new features are introduced or with increased effective model capacity. The training

error is typically zero, and the model fits the training data perfectly. Similar observations have also been made regarding statistical learning theory [30] and the generalization of a model in a highly overparameterized regime [13].

The discovery of double descent initially raised hopes that it would provide a comprehensive explanation for the performance of DNNs, given that double descent encompasses a wider range of phenomena than SLT. However, double descent is rarely observed in realistic training of DNNs, and it now seems unlikely that this phenomenon is an essential contributor to generalization in DNNs [28].

2.3 Sharpness in the weight landscape

Empirical evidence has shown that stochastic gradient models that converge to flatter minima tend to generalize better than those that converge to sharp minima. Among others, [31] defines a flat minimum as a wide region in weight space where every weight vector from that region results in a comparable error. To highlight this point, flat minima regions only require low precision weights compared to high precision for sharp minima. Flat minima are less complex, require fewer bits to describe and based on the minimum description (Message) length theory, generalize better [32].

2.3.1 Simplified flatness

Keskar et al. [33] empirically observed that large-batch methods are drawn to regions of sharp minima, which resulted in the degradation of a model’s ability to generalize. Conversely, small-batch methods are attracted to wider and flatter minima, as shown in Figure 2.3. The figure also shows that flat minima are insensitive to variations and provide robustness to random perturbations, an essential characteristic of a deep network. The low generalization performance of large batch methods can be attributed to the high number of positive eigenvalues in the Hessian matrix $\nabla^2 L(w^*)$.

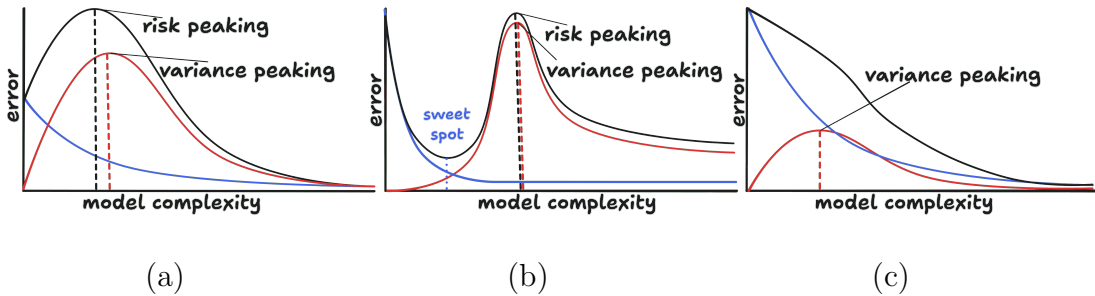


Figure 2.2: A plot of the test loss as the sum of the bell-shaped variance and monotonic bias. Case a) where the bias is much larger than the variance, resulting in a monotonically decreasing risk curve. Case b) where the variance and bias alternate in regime dominance, resulting in a double descent risk curve. Case c) where the variance is much larger than the bias, resulting in a bell-shaped risk curve that decreases after the threshold. Adapted from [25].

Methods with a few positive eigenvalues are known to generalize well, which contradicts the claims by Keskar et al. While this work primarily focused on the maximum loss within the restricted region of the minima, Chaudhari et al. [34] proposed the Entropy-SGD algorithm that utilizes Stochastic Gradient Langevin Dynamics (SGLD) to estimate the local entropy gradient. This compares the marginal likelihood of two Bayesian priors centred at a solution with small and large local-entropic loss. In essence, the local-entropic loss was constructed by connecting the Gibbs distribution and optimization problems. A Bayesian prior may result in a small marginal likelihood when centred on an isolated solution instead of one centred within a wide valley [35]. This also confirmed that local extrema with low generalization have the majority of small eigenvalues in the Hessian. Despite attempts by Keskar et al. [33] to remedy the sharp minima problems through data augmentation, conservative training and adversarial training, they could not overcome the problem.

2.3.2 A deeper look at flatness

The definition of flatness above does not entirely explain generalization due to the added complexity of deep models. For instance, in nonidentifiable rectified networks where some



Figure 2.3: A sketch of valleys with flat minima vs. sharp minima. This illustrates the high sensitivity of a training function to perturbations, which negatively affects generalization. Regions with flat minima are less complex and require low precision weights, making them likely to generalize better. Adapted from [33].

parameters in the model have little or no influence, there exist geometries resulting from symmetries created by these architectures that can be used to create comparable models with sharper minima [36]. In addition, directly associating Hessian-based sharpness to explain generalization is somewhat problematic because function reparameterization can drastically change and modify the parameters in the ReLU-MLP without changing the generalization properties [37].

Furthermore, sharpness, as defined in Section 2.3.1, cannot be utilized solely to determine the generalization behaviour of a deep neural network. It can be combined with the weight norm via Probably Approximately Correct (PAC)-Bayes [38] to obtain suitable complexity measures. PAC-Bayes bounds are always valid or time uniform for fixed sample sizes and also hold for stopping times [39]. The authors argue that the perturbation loss and empirical loss gap can be used as a sharpness metric.

The smoothness of model generalization can also be associated empirically with the local characteristics of a solution using the PAC-Bayes framework. This hypothesis was tested by choosing an ideal perturbation algorithm to optimize the perturbed empirical loss for superior model generalization. Instead of using heuristic-based perturbation approaches [40]–[42], Wang et al. [37] included the second order bound information from [35] for

better PAC-Bayes bound optimization level. This technique also illustrated some inherent regularization properties by improving generalization performance on unseen data.

2.3.3 Sharpness aware minimization

Recent work has shown an apparent relationship between the geometry of a loss landscape and a model’s generalization ability [23], [33], [43]. The complex and non-convex loss landscape may converge to a sharp minimum and generalize poorly. A novel approach known as Sharpness Aware Minimization (SAM) was introduced to address the challenge of poor generalization in deep neural networks [44]. SAM attempts to minimize both the loss value and loss sharpness simultaneously. It uses indiscriminate perturbed gradients to assist a model in exiting the sharp minimum when it reaches a sharp area. Furthermore, instead of using only parameters with low loss values, SAM finds and utilizes parameters found in valleys with consistently low loss values. In essence, results are formulated in a minimax optimization problem where gradient descent may be performed. However, SAM was characterized by numerous computational inefficiencies.

SAM needed two forward-backwards computations for each optimization, which made it computationally inefficient. Mi et al. [45] investigated Sparse SAM (SSAM) to address the computational costs associated with SAM and to improve model generalization. At its core, SSAM utilized Fisher information [46] and dynamic sparse training to determine the binary sparse mask used to find the parameters to be perturbed. SSAM empirically showed a flatter landscape contribution and better performance with only 50% of the parameters compared to SAM.

Other improvements to SAM include adaptive sharpness of loss function to remedy scale dependency complications for improved performance [47]. Efficient SAM was also proposed to provide efficiency without compromising generalization performance [48]. ESAM used efficient training methods such as stochastic weight perturbation and sharpness-sensitive data selection to improve the computational efficiency of standard SAM. To overcome SAM’s poor approximation error, Random SAM was proposed, which utilized

Gaussian noise in perturbation initialization to improve the inner maximization without introducing additional cost [49]. These improvements have yielded model improvements for various network architectures and datasets but have not entirely solved the generalization mystery since, no sharpness measure proposed to date consistently predicts generalization across tasks and architectures [50].

2.4 Margins

In ML research, margins are useful in determining the classifier’s certainty when predicting the correct label for a given input data: the further away a sample is from the decision boundary the more confidence or certainty is attached to the prediction. Margins may also assist in guiding the theoretical generalization bound and algorithm design in ML. Classical margin concepts have greatly contributed to the theoretical research and produced exceptional empirical results for classification [51] and regression [52]. However, these applications may only be constrained to shallow (typically one or two hidden layers) networks due to added complexity introduced by the overparameterized, nonlinear and high-dimensional modern deep neural networks and their decision boundaries. Therefore, the measurement of margins in such high dimensional spaces is intractable [53].

2.4.1 Linear considerations

Crammer et al. [54] highlighted two natural ways to describe an instance margin according to the classification rule. The simplest and most commonly encountered type is the sample margin, which computes the distance between a sample and the decision boundary created by the classifier. This distance is commonly referred to as the classification margin. It is important to note that several decision boundaries and distances may be suitable for specific datasets: several hyperplanes can separate a given data set but not all provide the largest separation. For instance, the Support Vector Machine (SVM) [55] aims to find the separating hyperplane with the largest sample margin, known as the maximum-margin

hyperplane, as shown in Figure 2.4a. The second alternative measure exists where the margin is calculated as the largest distance the sample points can move without altering the assigned label of the sample points, as shown in Figure 2.4b. This is referred to as the hypothesis margin and is primarily utilized in AdaBoost [56] along with the $l1$ -norm to measure the distance between hypotheses, which has demonstrated alignment to generalization properties in the linear model literature.

2.4.2 Non-linear considerations

The complexity of analyzing decision boundaries in high-dimensional deep networks has been addressed in various ways. [50], [53] adopted the first-order Taylor approximation to the gradient with respect to the weight to measure the distance of a point to the decision boundary. Unlike with [57], the approximation is applied to the input and hidden space instead of the output. Another proposition was utilizing the Jacobian function of the logits of the input (space) to define the margin at the input [58].

Research by Zhang et al. [13] showed that explicit regularisation alone was inadequate to influence the overall quality of model generalization. This prompted the question of how the generalization gap could be predicted using only training data and network parameters. Although generalization bounds approaches based on architectural complexity and noise stability had already been introduced, they were not empirically assessed in different practical settings [23], [57], [59], [60]. Jiang et al. [50] proposed a measure that strongly correlated to the generalization gap on selected test problems, which used the notion of margin in SVMs [55] and its extension to deep networks. Jiang et al. differ from the previous bounds and margin definitions because they proposed a normalization scheme that boosted prediction accuracy. Furthermore, they adopted margin distribution to cover the entire distribution distance instead of maximising a single margin. By so doing, they achieved better accuracy and used margin information across layers instead of just the output. Such measures can be applied to deep feedforward networks with any architecture and have assisted in finding loss functions that provide better generalization.

A recent contribution by Mouton et al. [61] highlighted that despite the correlation between margins and the generalization ability of a model from its hidden margins, there was a missing link between input margins and generalization. Mouton et al. showed the link by constraining the input margins to enhance generalization. The approach increased the predictive power, outperformed current hidden margin approaches and required no random layer normalization. The observed relationships between margin size and generalization were not related to network architecture in [61]; that is the direction pursued in the current work.

2.5 Expressivity and learnability

Various distinguishing factors between deep and shallow MLPs have been a topic of investigation for decades. Particular focus was put on the advantages of deep network function approximation and deducing shallow network limitations. Shallow networks were found to be incapable of localized approximation [62], limited lower bound approximation of smooth functions with high confidence [63] and their curse of dimensionality [64]. Deep networks, on the other hand, have proven to have localized approximation characteristics beyond the capability of shallow networks [65], [66]. The concept of expressivity in this context is premised around approximating functions expressed by deep neural networks but not by shallow neural networks with a comparable number of weights or neurons. Network expressivity tries to explain how neural networks can estimate functions based on their structural properties, such as width, depth and layer type [67]. It investigates the inherent nature of deeper neural networks, which exhibit capacities (scope of model’s function approximation) larger than those of shallow networks. In Lin et al. [68], deep neural networks were observed to have excellent expressive capabilities (on localized and sparse approximations) over shallow neural networks with similar parameter counts. Research by Bianchini et al. [69] also emphasizes that network depth is more effective on the implemented function complexity than its generalization capability. As a result, deep neural networks can express more complex functions without sacrificing their generalization ability compared to shallow neural networks for the same neurons.

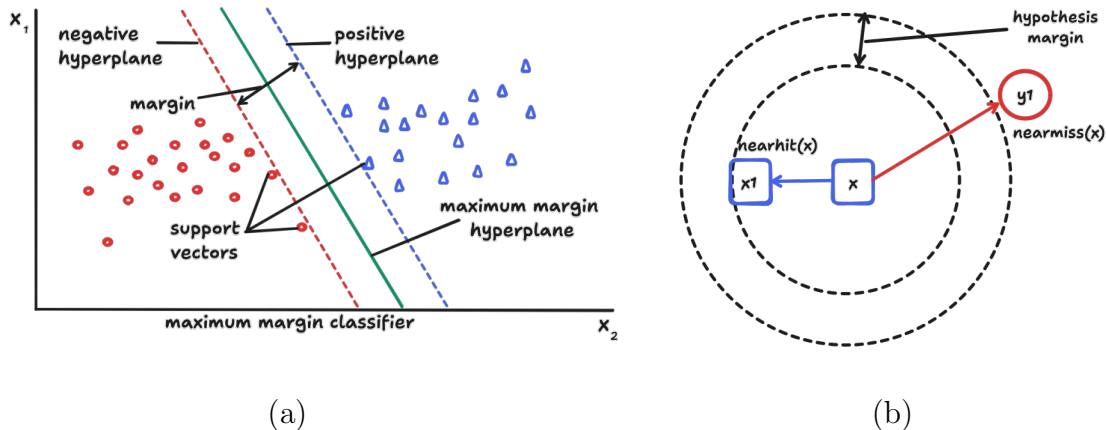


Figure 2.4: a) A sketch of the maximum margin classifier shows that positive and negative hyperplanes can separate the two classes but not with maximum margin. The maximum margin hyperplane maximises the separation from its closest point from each class. b) To compute the hypothesis margin at x for a set of points P , $\text{nearmiss}(x)$ and $\text{nearhit}(x)$ denote the closest point to x in P with different and same labels, respectively.

A related line of reasoning focuses on the learnability of various classes of networks [70]: even if two networks have similar expressive power, it may be that one is more likely to be trained successfully. In light of the large number of local minima in a typical network [71], such a distinction is not improbable. However, as discussed below, this does not seem to bear directly on the observed superiority of deep CNNs. In addition, the enhanced generalization of deep networks seems to require non-MLP architectures, such as a CNN or transformer [5], which renders these findings moot.

2.6 Implicit generalization and effective complexity

The independent variable in Fig. 2.1 is model complexity, and in conventional statistics, such complexity is straightforwardly represented by the number of parameters in a model. An alternative way to reconcile traditional bias-variance arguments with evidence such as that of [13] is to employ alternative definitions of complexity. This has been a fertile avenue for recent thinking about generalization [20]. A detailed review is available in [72]

which summarises some of the most important points.

The principal insight of this approach is that various factors combine to determine the complexity of a classifier in practice. These include the number of parameters, the model architecture, the training algorithm and the training data employed. For example, variants of stochastic gradient descent (typically used for training neural networks) tend to find solutions that are relatively close to the initial weight values. Thus, if these values can be shown to be of limited complexity by an appropriate measure, the existence of high-complexity networks elsewhere in weight space becomes irrelevant. If this is the case, the number of parameters is a misleading complexity measure.

Several alternative complexity measures have been proposed, including measures based on norms [73], sharpness in weight space [33] and robustness to small perturbations [74]. However, none of these measures lead to a simple relationship between complexity and generalization [23], similar to that in Fig. 2.1.

2.7 Geometric deep learning and interpolation

Modern DNNs are very effective in analysing and identifying hidden patterns within complex and large volumes of data. This is due to their ability to learn features automatically and achieve high model accuracy. For example, DNNs such as CNNs, used in computer vision and image processing, are excellent for extracting localized features by convolving a filter or kernel with an image through a striding window. DNNs have traditionally been applied to “Euclidean” signals such as speech, video and text. Since real-world data are not always in grid format, significant interest has been in expanding their application to non-Euclidean data [75], [76] such as social networks. This is a challenge since many standard data properties, such as shift-invariance, vector space structure and global parameterisation, are irrelevant in the expanded case [77]. As a result, mundane operations like convolution are not straightforward and clear-cut.

The introduction of Geometric deep learning GDL has bridged the gap between these deep

learning techniques and non-Euclidean data [78]. GDL is generally applied to manifold and graph data and utilises methods that include a geometric prior [77], which is exploited to improve model quality [79]. Although there is no direct interest in such input spaces, a brief mention is necessary since it employs some concepts (interpolation, non-Euclidean space) that are similar but distinct from usage in the thesis.

2.7.1 Interpolation variations

The concept of interpolation refers to a mathematical approach to predicting unknown values based on known observations [80]. Research communities often define interpolation differently based on the techniques applied [81]–[83]. In deep learning, various forms of interpolation have been utilized in recent years [84]–[87]. For example, the definition used by Balestrieri et al. [88] suggests that interpolation is assured if the sample of interest lies within the convex hull of the dataset. This definition is derived from low-dimensional intuition and, as a result, experiences difficulties in high-dimensional spaces.

In contrast, our definition in Chapter 4 interpolates a sequence of elements between two points that least change the output. This approach does not depend on the dimensionality of the input and, therefore, does not rely on low-dimensional intuitions. Additionally, unlike Balestrieri et al., our approach does not suffer from exponential data requirements. Our interpolation is exclusively implemented in input space and not in latent space. Similar work, generating images from the underlying network for embeddings, has been achieved using autoencoders [89]; our goal, however, is not to generate novel images but to understand how different network architectures generalize around training data points.

2.8 Discussion

This chapter reviews factors that affect the generalization ability of DNNs. Classical considerations of bias-variance trade-off in Section 2.2, where a model’s complexity determined how well it would perform on unseen data were explored. However, this dominant

dogma contradicted the trend shown by modern neural networks, which are complex and overparameterized but tend to generalize well. Classically, models that can train well with relatively few parameters are likely to generalize more successfully since their parameters are constrained with lower levels of variance across task-irrelevant variations. This explanation formed the core of the so-called SLT but was never entirely satisfactory as a comprehensive explanation of generalization.

Another factor highlighted was the role of sharpness in weight space, where Stochastic Gradient Descent (SGD) models showed better generalization results due to their tendency to converge to flatter minima. These wider valleys provide higher insensitivity to training function perturbations, enhancing performance on unseen data. However, as a measure, sharpness could not be utilized as the sole indicator of generalization behaviour in DNNs. Section 2.4 highlighted that the relationship between margin size and generalization was unrelated to network architecture by constraining the input margin. Section 2.5 emphasized that deeper networks had superior expressive abilities on complex functions over their shallower counterparts without sacrificing their generalization ability. Despite the superior approximation properties, the expressivity of DNNs networks does not guarantee good generalization and trainability [90]. This is because overparameterized shallow networks can also fit training data perfectly and generalize well for certain problems. Section 2.6 examined the relationship between model complexity and generalization, where cases can exist where the number of model parameters may be a misleading complexity measure. Alternative definitions of complexity have recently been of much interest. However, these approaches have not proven effective in relating generalization directly to model complexity. Section 2.7 highlights the differences between our reference to the Euclidean and non-Euclidean space and the contrasting data types. Lastly, Section 2.7.1 contrasts our definitions of interpolation with others in the literature. It also discussed the importance of highlighting the space in which the research operated to distinguish between previous approaches in latent space.

Although the factors discussed above affect the generalization ability of ReLU networks, they are not an indicator or predictor of how well a given network will perform on unseen data. It has become evident that historical concepts such as SLT have little explanatory

power for modern ML systems. However, no competing approach to generalization has found widespread acceptance, and generalization is often treated as a somewhat mysterious property possessed by some systems but not others. It is still an open research area, and the rest of the thesis develops a simple framework for studying generalization in CNNs.

Chapter 3

Benefits of depth in CNNs

This chapter presents empirical investigations of the effect of depth on the generalization ability of standard deep CNNs, as guided by the concepts introduced in the literature. The results are not new but provide us with a simple and well-defined experimental framework.

3.1 Overview

In this chapter, we investigate the effects of depth in standard DNNs, specifically deep CNNs. This is primarily informed by the research in Davel et al. [91], which identified certain regularities in class-related activation patterns of ReLU-activated neural networks. This was observed in fully connected MLPs and provided helpful insight into the generalization abilities of DNNs. Davel et al. suggest that the cooperation of distinct classifiers formed by individual neurons results in redundancy and robustness, which increases if more network parameters are available. This stands in contrast to the classical bias-variance trade-off, discussed in Section 2.2, which posits that excess parameters beyond a certain number will decrease the accuracy of network generalization. Another regularity observed in feedforward neural networks of adequate depth was that class discrimina-

tion tends to shift to earlier layers. By mapping pixels into feature space, Zeiler et al. [92] observed the hierarchical nature of CNN layers where early layers respond to simple features and later ones to more complex features. These observations may provide an essential hint on the differences between CNNs and MLPs and thus will be central to our investigations.

The regularities observed in [91] did not suggest a significant advantage in performance when the depths of MLPs are adjusted. For CNNs, in contrast, there are many reports that additional hidden layers produce improved generalization. As a basis for our further investigations, we first wish to establish a simple CNN architecture where these advantages of greater depth can be systematically observed. In subsequent chapters, we will then strive to understand the causes of these improvements.

Section 3.2 describes the architectural design of the networks used to conduct the experiments. These design approaches are largely informed fundamentally by the work presented in [12], [16], [92]–[94]. Section 3.3 explores the attributes of MNIST, Fashion MNIST and CIFAR-10 datasets. Section 3.4 discusses the CNN performance on the abovementioned datasets at varying depth levels.

3.2 Experimental setup and generalization

We start by building a straightforward experimental setup to investigate the generalization behaviour of deep CNNs. We explore multiple architectures, with increasing depth, to conduct our training process. Figures 3.1 and 3.2 represent lower capacity networks guided by the input dimensions of the datasets used. Higher capacity networks, shown in Figures A.3 and A.4, double the number of neurons for every layer. Each feature extraction section of the network consists of a single convolution layer with a ReLU [94] activation function and a pooling layer. To ensure that we keep the setup as simple as possible, we have omitted any explicit regularization, such as batch normalization and dropout. The number of filters per convolution layer, optimizer, learning rate and batch size were set as hyperparameters to optimize the learning process. We have also adopted small

kernel sizes (3x3) with a unit stride as described by [16]. Each feature map first passes through a (2x2) pooling layer to summarize the most prominent features and reduce dimensionality. All architectures used a single fully connected layer of 20 neurons to combine the flattened output of the feature extraction section of the network. The fully connected layer outputs were fed into the output layer. For its simplicity and probabilistic interpretation, a Softmax logistic function layer was used at the output to normalize the probability distribution for the predicted classes.

In addition, our setup included comparative lower and higher capacity MLP networks, with 512 and 1024 neurons, respectively. As discussed in Section 2.5, the MLP networks were constructed with a single hidden layer of varying width depending on the set capacity. We also used a standard architecture, shown in Figure 3.3, without explicit regularization and a Softmax logistic function at the output. All architectures were optimized to reduce a cross-entropy loss function for each training sample batch with varying sizes as hyperparameters.

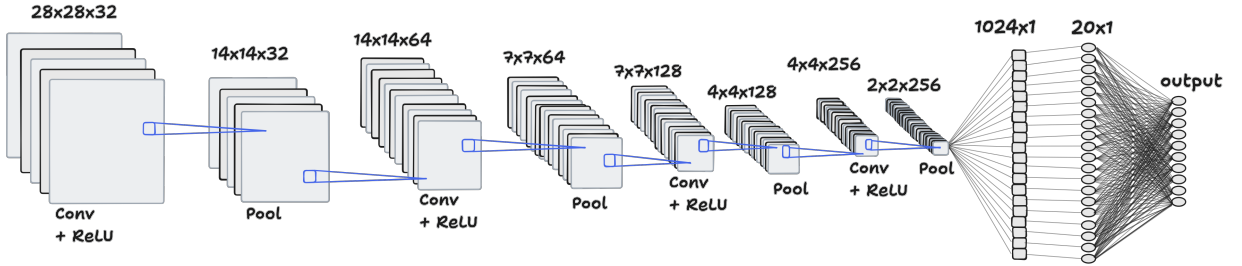


Figure 3.1: An architectural design of a lower capacity deep CNN with average pooling and a single 20-neuron fully connected layer. The network depth of 4 layers was only constrained by the input dimensions. Note that the dimensionalities shown are for FMNIST and MNIST datasets.

3.3 Data

Our experiments were conducted on the MNIST [93], Fashion MNIST [95] and CIFAR-10 [4] datasets. The datasets are standard, freely available and widely used as a point of

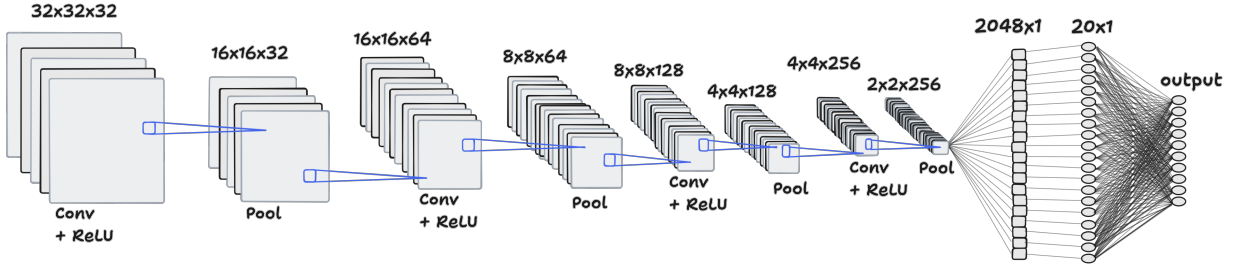


Figure 3.2: An architectural design of a lower capacity deep CNN with average pooling and a single 20-neuron fully connected layer. The network depth of 4 layers was only constrained by the input dimensions. Note that the dimensionalities shown are for the CIFAR-10 dataset.

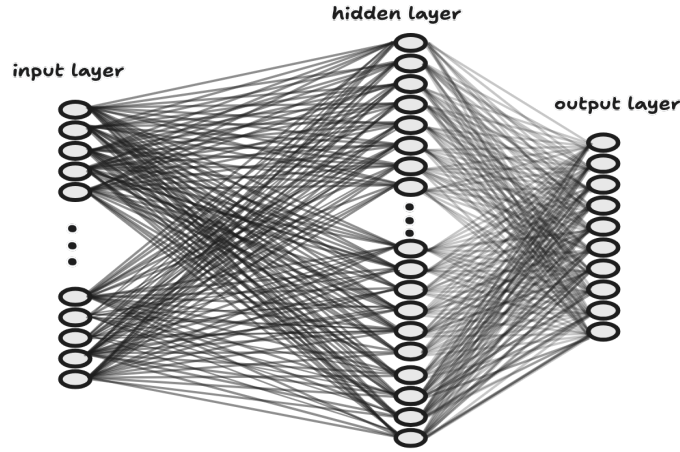


Figure 3.3: An architectural design of an MLP network. Lower capacity networks utilised 512 neurons, while higher capacity networks utilised 1024 neurons.

reference for computer vision tasks in ML. Although both these datasets are easy to recognize with state-of-the-art systems, they are sufficiently challenging to demonstrate the properties of the relatively simple CNNs that are our focus. This allows us to conduct experiments on relatively affordable hardware and prevents us from being distracted by many of the complications (such as data augmentation) required for state-of-the-art performance. Of course, this approach only makes sense if the lessons learned on these relatively simple systems and datasets also transfer to more competitive systems. We return to this issue in the conclusion section of our work.

MNIST is a dataset of real-world handwritten digits that range from 0 to 9. It has 70,000

grayscale image samples with 28x28 input features, each of which 60,000 were used for training and 10,000 for testing. There are 10 distinct classes of almost equal size, as shown in Figure 3.4a. Each image pixel value was normalized and re-scaled to a range between 0 and 1 to ensure that all input feature vectors have similar data distribution for faster network training. MNIST class memberships, shown in Figure A.1a, show a slight imbalance in training and test sets. The Fashion MNIST dataset, shown in Figure 3.4b, is similar to MNIST in that it has the same number of samples, input features, and 10 distinct classes and is also in grayscale. Fashion MNIST was created as a more challenging option to the MNIST dataset. Unlike MNIST, all the different classes are equally represented and there is no class imbalance, as shown in Figure A.1b. CIFAR-10 was the third dataset comprising 60,000 real-world object colour image samples. It has 50,000 colour (RGB) image samples with 32x32x3 input features each for training and 10,000 for testing. There are also 10 distinct classes, each with precisely 5,000 colour image samples, as shown in Figure 3.5. All class members in the training and test sets are balanced, as shown in Figure A.2. The normalization process was the same for all the datasets; no data augmentation was used.

3.4 Results

We started our experiments with a single convolutional layer of 32 neurons for lower capacity networks and 64 for higher capacity networks. Following standard CNN architecture design, all subsequent layers had double the number of neurons from the previous layer. The network depth was limited to a maximum of four layers because, after the last pooling layer, the spatial dimensions of the feature maps were reduced to a point where further layers would not provide meaningful processing. The general architecture remained the same for all the datasets but required different hyperparameters. No explicit regulariser was used to train the CNN and MLP models. lower and higher capacity models reached close to 0% training classification error but were saturated after several epochs. MNIST and FMNIST models displayed similar validation error curves for lower and higher capacity networks, as shown in Figures 3.6 and 3.8, respectively. The CIFAR-



Figure 3.4: a) MNIST and b) Fashion MNIST single-channel class dataset samples. Class members are grouped vertically for all 10 classes.

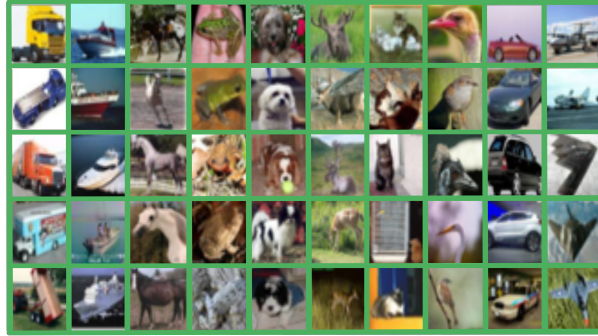


Figure 3.5: CIFAR-10 3-channel class dataset samples. Class members are grouped vertically for all 10 classes.

10 model output plots in Figure 3.9 show convergence around 50 epochs but display more distinct validation error curves between the network depths than MNIST and FMNIST.

The MLP networks were also trained on the standard architecture, with 512 and 1024 neurons for lower and higher capacity, respectively. MNIST and FMNIST models were also trained to 0% training classification error for lower and higher capacity networks, while CIFAR-10 could only reach 0% training classification error for higher capacity networks. We selected the best five trained models from both MLPs and CNNs and calculated the standard error to show the consistency of the training process

Once the training process converged, we observed a gradual decrease in validation loss for both lower and higher capacity networks for the MNIST dataset in CNNs. However,

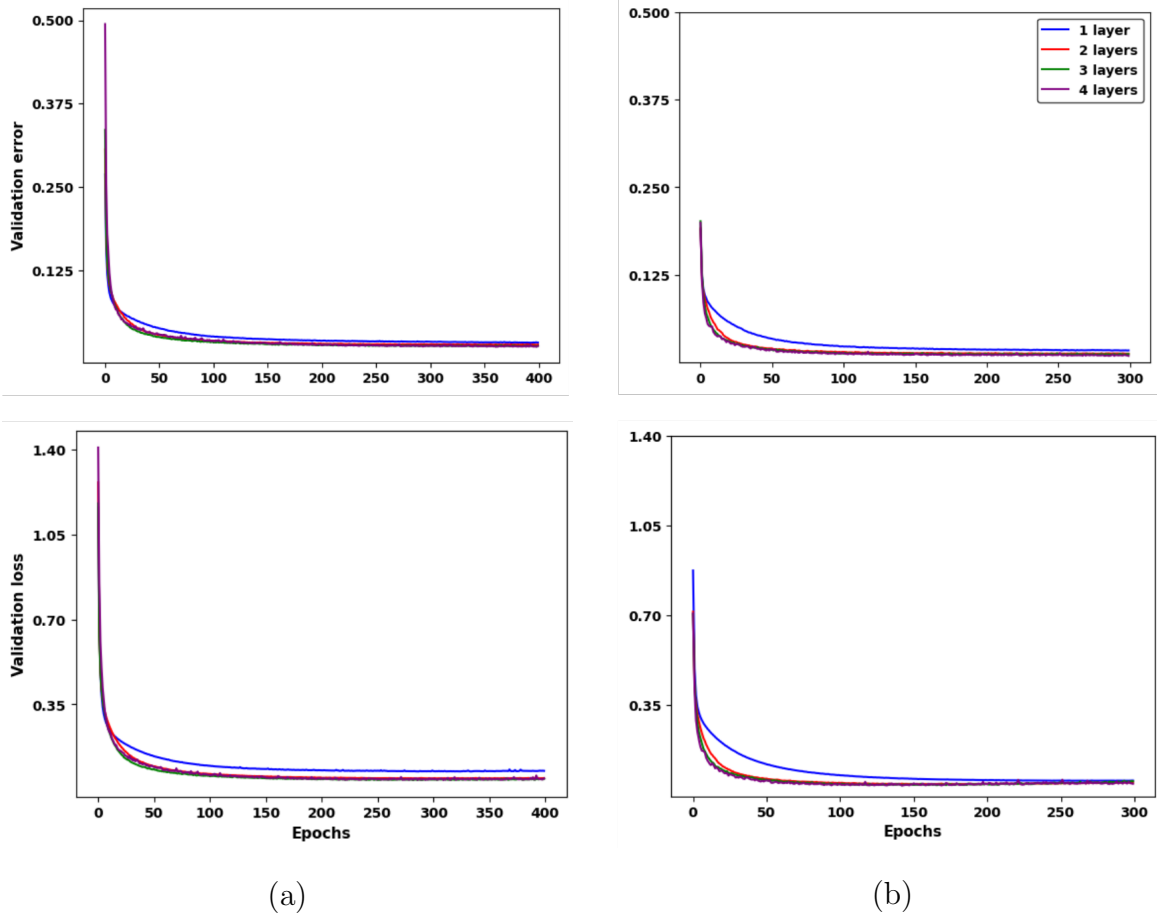


Figure 3.6: Validation error rate and loss per CNN network depth for MNIST dataset. a) represents a lower capacity network that starts from 32 neurons in the first layer and doubles in subsequent layers. b) represent a higher capacity network that starts from 64 neurons and doubles in subsequent layers.

the validation loss curves for the FMNIST and CIFAR-10 datasets, with greater depths, show a U-shaped performance curve up to 100 epochs. It is also evident from Figure 3.9b that the U-shape is achieved earlier in the training process and is also much sharper than in Figure 3.9a. Additionally, there is no evidence of a distinct risk peak and gradual decrease of the validation error after some threshold, as shown in Figure 2.2b. Our observations also show that the deeper networks for the MNIST dataset have a lower validation loss than shallower ones. This is generally the opposite behaviour displayed by the FMNIST and CIFAR-10 networks. The MLP model results, shown in Figure 3.7, exhibit a consistent trend in the validation error and loss for all the datasets. The

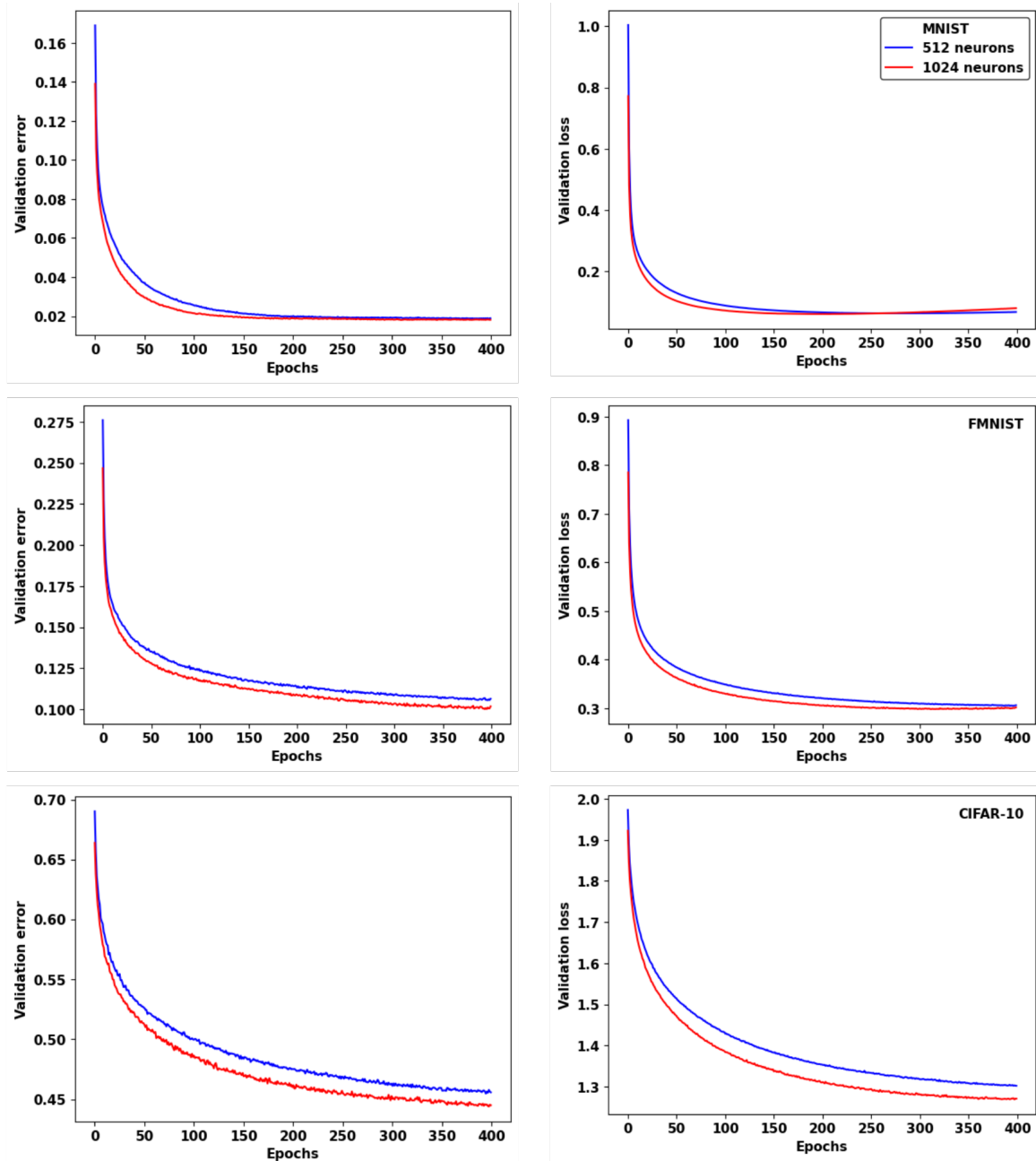


Figure 3.7: Validation error and loss per MLP network capacity for MNIST, FMNIST and CIFAR-10 datasets. 1024 neurons represent higher capacity, while 512 neurons represent lower capacity models.

validation error and loss increased as the dataset became more challenging. In addition, the higher capacity network performed slightly better than the lower capacity model for all datasets.

Tables 3.1 and 3.2 summarise each MNIST dataset’s lower and higher capacity model performance, respectively. The networks showed good validation error rates on unseen data. FMNIST datasets, shown in Table 3.3 and 3.4, higher capacity networks outperformed the lower capacity networks. Additional filters to each layer improved the generalization performance of the model. However, there was a slight increase in validation loss for specific depths. The deeper the network, the better the overall validation error rate. This observation was consistent for both lower and higher capacity networks. Shallower networks for the CIFAR-10 dataset could not fit the training data due to insufficient capacity, shown in Table 3.5 and 3.6. As a result, higher capacity networks produced the best results and met our requirement to train models to almost 0 % training error. This systematic benefit of network depth stands in sharp contrast to the behaviour of MLPs, where empirical evidence in Table 3.7 shows that shallow networks closely approximate the performance of optimal deep networks if hidden layer sizes are chosen appropriately.

CNN layers	Filters	Trainable params	Validation error	
			Average error	Standard error
0	512	407,050	1.77	0.022
1	32	125,990	1.73	0.070
2	32+64	81,766	1.43	0.024
3	32+64+128	133,862	1.22	0.045
4	32+64+128+256	408,550	1.19	0.071

Table 3.1: Average lower capacity MLP (represented by 0 CNN layers) and CNN performance results for increasing depth on the MNIST validation sets. Each number in the filters column represents the number of neurons per subsequent convolutional layer. Note that the number of neurons in the fully connected layer in CNNs was set to 20 for all experiments. Validation error was averaged over 5 trained models and standard error represents how spread out the values are around the mean.

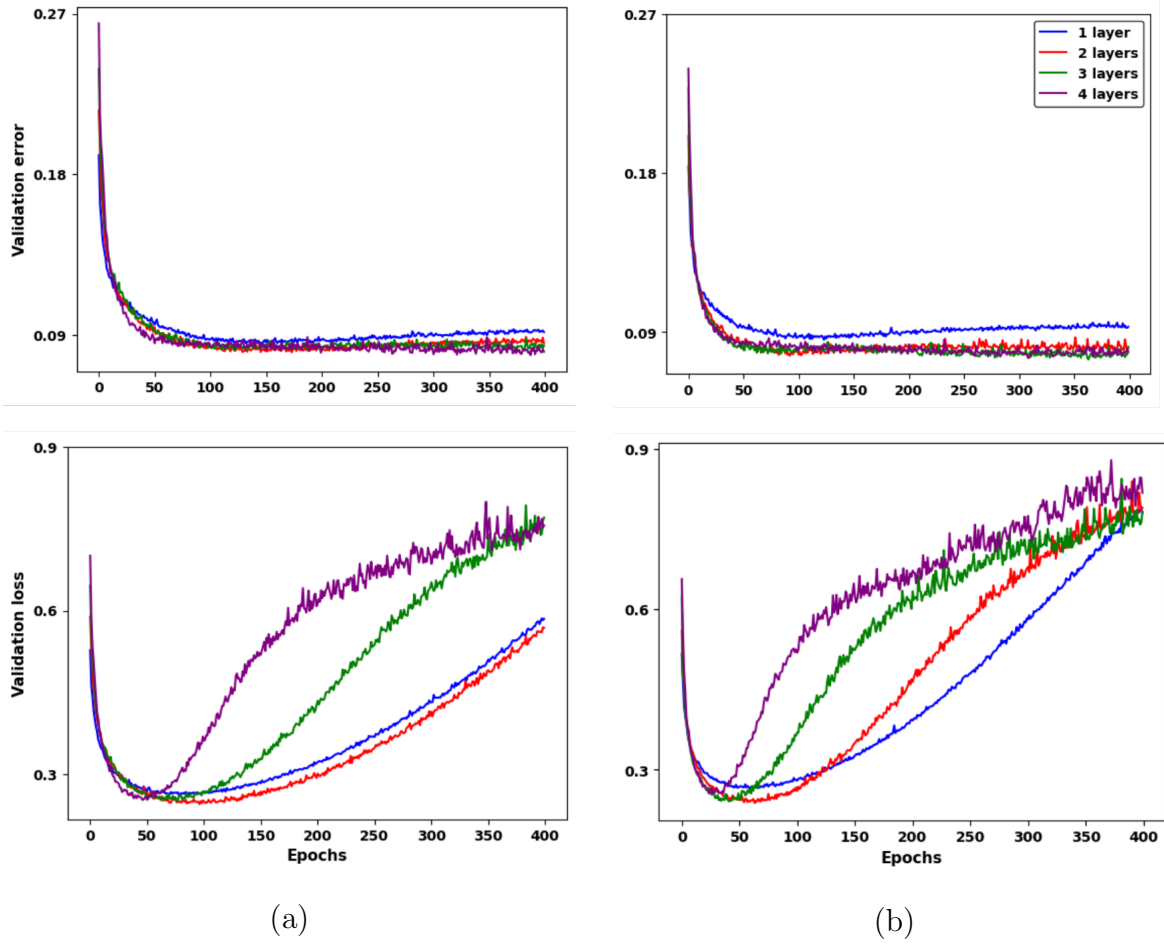


Figure 3.8: Average validation error rate and validation loss per CNN network depth for FMNIST dataset. a) represents a lower capacity network that starts from 32 neurons in the first layer and doubles in subsequent layers. b) represent a higher capacity network that starts from 64 neurons and doubles in subsequent layers. Note that 5 models were averaged to generate the plots.

3.5 Discussion

This chapter presents the importance of depth in the generalization ability of standard CNNs. Our findings confirm and align with some concepts highlighted in the literature section. However, our results did not concur with the classical bias-variance trade-off, which suggested a decrease in generalization due to network overparameterization. We first explored standard MLPs and multiple CNN architectures with increasing depth and capacity. The observed systematic improvements in model performance indicated that

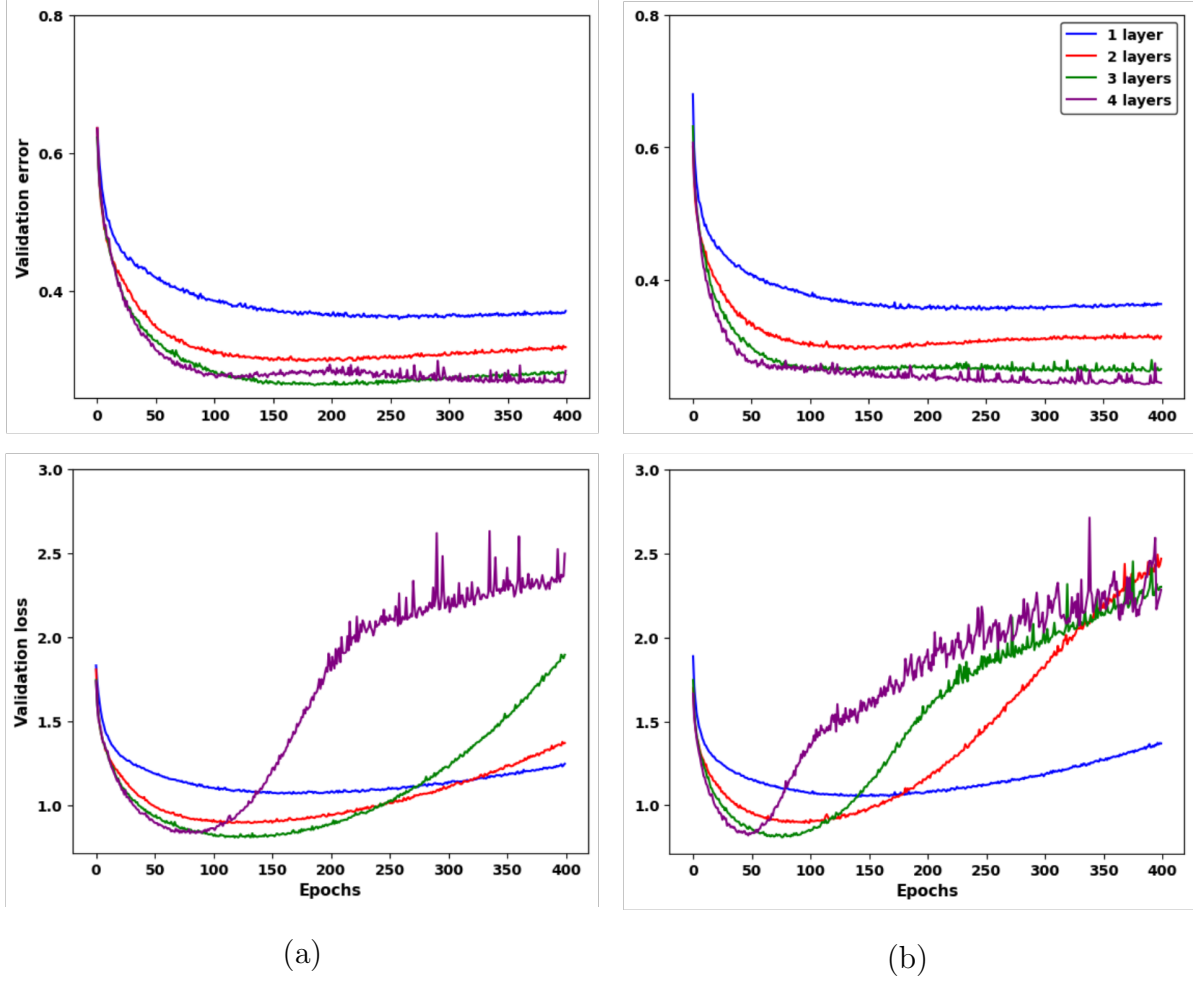


Figure 3.9: Average validation error rate and loss per CNN network depth for CIFAR-10 dataset. a) represents a lower capacity network that starts from 32 neurons in the first layer and doubles in subsequent layers. b) represent a higher capacity network that starts from 64 neurons and doubles in subsequent layers. Note that 5 models were averaged to generate the plots.

deeper CNNs performed better than their shallower counterparts. Firstly, unlike MLPs, additional network width did not significantly improve generalization. Secondly, the networks did not utilize any explicit regularization. These factors indicated that it would be reasonable to attribute the superior model performance to network depth. Lastly, the increase in network capacity as depth increased did not degrade the generalization ability of the networks as the classical bias-variance trade-off would suggest. In fact, our networks fit the training set close to 100% and generalized well, indicating that the classical

Layers	Filters	Trainable params	Validation error	
			Average error	Standard error
0	1024	814,090	1.74	0.028
1	64	251,750	1.695	0.046
2	64+128	200,166	1.255	0.046
3	64+128+256	451,814	1.208	0.028
4	64+128+256+512	1,591,014	0.959	0.036

Table 3.2: Average higher capacity MLP (represented by 0 CNN layers) and CNN performance results for increasing depth on the MNIST validation sets. Each number in the filters column represents the number of neurons per subsequent convolutional layer. Note that the number of neurons in the fully connected layer in CNNs was set to 20 for all experiments. Validation error was averaged over 5 trained models and standard error represents how spread out the values are around the mean.

theory may not fully apply to modern networks. The subsequent chapters will build on the results in this section and form a basis for quantifying the generalization differences attributed to the different CNN depths.

CNN layers	Filters	Trainable params	Validation error	
			Average error	Standard error
0	512	407,050	10.41	0.055
1	32	125,990	9.428	0.249
2	32+64	81,766	8.572	0.064
3	32+64+128	133,862	8.408	0.167
4	32+64+128+256	408,550	8.165	0.100

Table 3.3: Average lower capacity MLP (represented by 0 CNN layers) and CNN performance results for increasing depth on the FMNIST validation sets. Each number in the filters column represents the number of neurons per subsequent convolutional layer. Note that the number of neurons in the fully connected layer in CNNs was set to 20 for all experiments. Validation error was averaged over 5 trained models and standard error represents how spread out the values are around the mean.

CNN layers	Filters	Trainable params	Validation error	
			Average error	Standard error
0	1024	814,090	10.06	0.495
1	64	251,750	9.482	0.150
2	64+128	200,166	8.159	0.164
3	64+128+256	451,814	7.808	0.098
4	64+128+256+512	1,591,014	7.708	0.035

Table 3.4: Average higher capacity MLP (represented by 0 CNN layers) and CNN performance results for increasing depth on the FMNIST validation sets. Each number in the filters column represents the number of neurons per subsequent convolutional layer. Note that the number of neurons in the fully connected layer in CNNs was set to 20 for all experiments. Validation error was averaged over 5 trained models and standard error represents how spread out the values are around the mean.

CNN layers	Filters	Trainable params	Validation Error	
			Average error	Standard error
0	512	407,050	49.04	0.45
1	32	125,990	37.04	0.41
2	32+64	81,766	32.65	0.85
3	32+64+128	133,862	28.12	0.31
4	32+64+128+256	408,550	27.94	0.94

Table 3.5: Average lower capacity MLP (represented by 0 CNN layers) and CNN performance results for increasing depth on the CIFAR-10 validation sets. Each number in the filters column represents the number of neurons per subsequent convolutional layer. Note that the number of neurons in the fully connected layer in CNNs was set to 20 for all experiments. Validation error was averaged over 5 trained models and standard error represents how spread out the values are around the mean.

CNN layers	Filters	Trainable params	Validation error	
			Average error	Standard error
0	1024	814,090	47.28	0.42
1	64	251,750	36.50	0.39
2	64+128	200,166	31.33	0.56
3	64+128+256	451,814	26.40	0.51
4	64+128+256+512	1,591,014	24.21	0.64

Table 3.6: Average higher capacity MLP (represented by 0 CNN layers) and CNN performance results for increasing depth on the CIFAR-10 validation sets. Each number in the filters column represents the number of neurons per subsequent convolutional layer. Note that the number of neurons in the fully connected layer in CNNs was set to 20 for all experiments. Validation error was averaged over 5 trained models and standard error represents how spread out the values are around the mean.

		MNIST		FMNIST		CIFAR-10	
MLP Layers	Filters	Avg	Std	Avg	Std	Avg	Std
1	1024	1.93	0.019	11.32	0.05	45.25	0.27
2	768+256	1.91	0.014	11.46	0.04	45.15	0.15
2	1024+512	1.82	0.013	11.40	0.06	45.63	0.12
3	640+256+128	1.93	0.029	11.29	0.05	45.76	0.12
3	1024+512+256	1.89	0.031	11.04	0.04	45.26	0.13
4	576+256+128+64	1.99	0.029	11.44	0.08	46.08	0.14
4	1024+512+256+128	1.91	0.038	10.93	0.07	45.59	0.17

Table 3.7: Average MLP performance results for increasing depth on the MNIST, FMNIST and CIFAR-10 validation sets. Average validation error (Avg) and standard error (Std) thereof computed over 5 trained models starting from different random initializations.

Chapter 4

Piecewise linear model

In this chapter, we develop tools to investigate the superior generalization of deep CNNs, and use these tools to discover a systematic trend in network behaviour that explains this superiority.

4.1 Overview

Our experiments have shown that adding more convolutional layers improves model generalization. This empirical evidence indicates that deep and large-capacity networks perform better than their lower capacity counterparts. These networks seem to create more accurate classifiers, and as we increase the number of layers, they seem to cope more successfully with distortions or variations in the input space. CNNs, through the combination of convolution and pooling, appear to automatically create mechanisms that ensure that two things are judged as similar when they are visually similar. The more capacity the network has, the more flexible this similarity metric seems.

The empirical evidence provided in Section 4.6 confirms the conclusions of [13]. This is in

direct contrast to the classical learning theory and its key concept of bias-variance trade-off, which would imply that over-parameterized, high-dimensional and high-complexity networks should not perform well on unseen data. Despite the clear experimental evidence, there is currently no comprehensive explanation of the observed pattern of out-performance (which strongly favours deeper networks).

In Section 4.2, we take an alternative approach to SLT to understand the generalization capabilities of deep CNNs. We do this by introducing the concept of classifiers as converters. This approach motivates us to focus on the locally linear mapping computed by deep CNNs to understand the properties of CNN-based converters in Section 4.3. In Section 4.4, we develop the precise theoretical expression of the linear representation of a convolution layer. Section 4.5 investigates the linear interpolation between training samples from the same class by networks of varying depth, using the linear representation to probe the properties of such converters in order to gain insight into the benefits of greater network depth.

4.2 Classifiers as converters

We propose to view classifiers as converters between the following two metric spaces.

- i The input or feature space, and
- ii Output space which reflects some measure of the likelihood that each of the target classes occurred (e.g. the class posterior probabilities)

Naive intuitions about metric spaces are heavily influenced by our everyday experience of the three spatial dimensions. Most prominently, these intuitions assume a near-Euclidean metric in a typical space after appropriate scaling. For the output space, that intuition is helpful in that one expects the outputs of a well-trained classifier to be concentrated on the unit vectors representing each of the classes in the standard one-hot encoding. Additionally, the Euclidean distance from any of these vectors reasonably indicates how

improbable the corresponding class is.

The input space, however, behaves quite differently. Consider the binary images in Figure 4.1, if the brightness of each pixel is treated as a distinct dimension in input space, images in Figure 4.1b and Figure 4.1c are at the same Euclidean distance from the image in Figure 4.1a but in a metric that corresponds to the semantics of the image, they are quite different. Therefore, we can ask how a successful classifier transforms the highly distorted metric at the input to the more regular metric at the output.

This seems like a fruitful way to investigate DNNs, since such problems with severely distorted input metrics are precisely those for which these classifiers generalize with extraordinary accuracy. For more *regular* input spaces, such as those defined by conventional feature extraction, DNNs have not been shown to perform particularly well [96], [97].

One way to understand the relationship between the input and output metrics is to study their respective invariances under the semantic interpretation. If the output variables approximate posterior probabilities, their values will approximately sum to 1, and thus, for C classes, there are $C - 1$ independent directions of variation and a single invariance. On the input side, matters are much more complicated. Consider, for example, an input image of size $D = N \times M$ pixels. If we were to perform a linear mapping from input to output, the $C - 1$ dimensions of class variation in the input space would generically map to a $C - 1$ dimensional subspace of the input space. Thus, there would be $D - C + 1$ dimensions, which are invariants for classification. The same relationships hold locally for a network with a piecewise-linear transfer function, such as an MLP or CNN with ReLU non-linearities. Within each linear patch of the piecewise-linear input space, there will be $D - C + 1$ class-invariant dimensions and $C - 1$ dimensions that span the class variations.

We have previously shown (see Section 3.4) that CNN architectures with different numbers of convolutional layers differ systematically in how well they generalize. We propose to study the source(s) of those differences by investigating these two local subspaces for different CNN architectures.

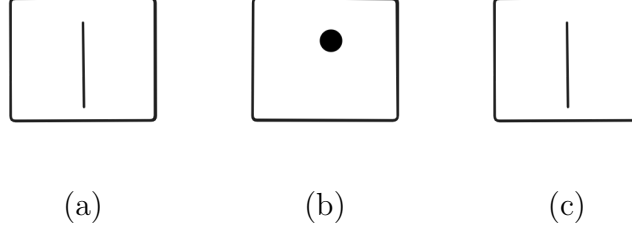


Figure 4.1: Image (c) is shifted one pixel to the right compared to image (a); hence, the respective Euclidean distances between images (b) and (c) to image (a) are the same.

4.3 Input-output mapping in a CNN

We first express the convolution operation as a conventional matrix-vector product to find the local linear mapping computed by a CNN. Let us denote the activities of the nodes in layer i , channel k of a CNN by the vector $\mathbf{x}_k^i$, where the rows of an image plane or feature map are concatenated to form a vector. These vectors can, in turn, be concatenated to form a vector $\mathbf{x}^i$ containing all the activities of layer i . As shown in Section 4.4 below, the convolution and pooling operations in one layer can be **locally** combined into a single linear operation

$$\mathbf{x}^{i+1} = \mathbf{W}^i \mathbf{x}^i, \quad (4.1)$$

where $\mathbf{W}^i$ is a matrix that depends on the convolution filters, biases, the activations of the ReLUs, and the pooling function in layer i (details below). The output of the final convolution layer is therefore

$$\mathbf{x}^{L-1} = \prod_{i=0}^{L-1} \mathbf{W}^i \mathbf{x}^0. \quad (4.2)$$

Any fully connected layers are similarly locally linear so that a single matrix-vector product can locally represent the mapping from input to logits. As will become clear below, we do not need to include the softmax operation in our representation, so our focus will be on the mapping

$$\mathbf{z} = \mathbf{W}(\mathbf{x}^0) \mathbf{x}^0, \quad (4.3)$$

where $\mathbf{z}$ is the vector of logits and $\mathbf{W}$ is a product of the matrices representing the convolution layers as well as the fully connected layers, which depend on the input $\mathbf{x}^0$ according to the derivation in Section 4.4.

We aim to use this representation to probe the input-output mappings of CNNs with different levels of generalization. One way to do this is to visualize the neighbourhoods adjacent to representative training samples. We select a sample $\mathbf{x}^{0c}$ from class c , and then trace out the way the input changes as we move directly in the directions of each of the other classes c' . Since we know that the class-variable subspace contains $C - 1$ dimensions, these $C - 1$ variations form a complete linear basis for the local between-class variability of the input space. We compute these variations in detail using the following procedure.

1. Select two classes c and c' , also, a representative sample $\mathbf{x} = \mathbf{x}^{0c}$ (i.e. a sample with high correct posterior probability in the trained classifier) from class c .
2. Compute the mapping matrix $\mathbf{W}(\mathbf{x})$ centred on $\mathbf{x}$. By construction, it yields the output logits $\mathbf{z} = \mathbf{W}\mathbf{x}$.
3. Move in a straight line between this sample and the closest location classified as c' . Since the rows of $\mathbf{W}(\mathbf{x})$ map from the input space to the class logits, this motion is

$$\mathbf{x} \rightarrow \mathbf{x} + \epsilon(\mathbf{W}_{c'} - \mathbf{W}_c), \quad (4.4)$$

where $\mathbf{W}_i$ is the i -th row vector of $\mathbf{W}$.

4. Output this image and stop the process if the posterior of c' exceeds that of c . Otherwise, repeat the process from Step 2.

This process produces images representing one class variability dimension around $\mathbf{x}^{0c}$. By repeating it for all possible choices of c' , we should be able to gain a reasonable intuition of the types of invariances captured by the network at that location, and further investigation for all choices of c will enhance that intuition.

Alternatively, we can use a similar process to investigate the approximately invariant space between samples that belong to the same class. Such an investigation may suggest a metric of intra-class invariance that predicts how well a given network will generalize, as shown below.

4.4 Linear representation of a convolution layer

A minimal convolution block in a standard CNN consists of three major stages:

1. Each input plane $\mathbf{x}_k^i$ is convolved with the relevant kernel $\mathbf{f}_{kk'}^i$ (i and k are plane and spatial indices, respectively):

$$\mathbf{y}_k^i = \sum_{k'} \mathbf{x}_{k'}^i \mathbf{f}_{kk'}^i. \quad (4.5)$$

2. The results of the convolution are passed through a ReLU non-linearity:

$$z_{k,ab}^i = \max(y_{k,ab}^i, 0), \quad (4.6)$$

where the indices a, b refer to the coordinates of the feature map.

3. Finally, the inputs of the next layer are computed by pooling the relevant z values with an appropriate operator, either maximum or average pooling.

$$\mathbf{x}_k^{i+1} = \text{Pool}(\mathbf{z}^i). \quad (4.7)$$

To implement the convolution as a matrix-vector product, we concatenate the images or feature maps on all channels into a single vector, as described above. The convolution is achieved by inserting the relevant kernel value for each product in the corresponding matrix position, for example, row $r = k \times D^i + d$ produces the output at position d of the k -th channel and therefore consists of the concatenation of the elements of $\mathbf{f}_{kk'}^i$, offset by d positions and correctly spaced. This matrix we denote by $\mathbf{V}^i$. For a 5x5 input, 3x3 kernel, two input channels and two output channels, for example, the structure of $\mathbf{V}^i$ is as follows, assuming zero padding and same convolution (where the dimensions of the input and output are equal).

$$\begin{bmatrix} f_{00,11}^i & f_{00,12}^i & 0 & 0 & 0 & f_{00,21}^i & f_{00,22}^i & 0 & \cdots & 0 & f_{01,11}^i & f_{01,12}^i & \cdots & \cdots \\ f_{00,10}^i & f_{00,11}^i & f_{00,12}^i & 0 & 0 & f_{00,20}^i & f_{00,21}^i & f_{00,22}^i & 0 & \cdots & f_{01,10}^i & f_{01,11}^i & f_{01,12}^i & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ f_{00,01}^i & f_{00,02}^i & 0 & 0 & 0 & f_{00,11}^i & f_{00,12}^i & 0 & \cdots & 0 & f_{01,01}^i & f_{01,02}^i & \cdots & \cdots \\ f_{00,11}^i & f_{00,12}^i & 0 & 0 & 0 & f_{00,21}^i & f_{00,22}^i & 0 & \cdots & 0 & f_{01,11}^i & f_{01,12}^i & \cdots & \cdots \\ f_{00,10}^i & f_{00,11}^i & f_{00,12}^i & 0 & 0 & f_{00,20}^i & f_{00,21}^i & f_{00,22}^i & 0 & \cdots & f_{01,10}^i & f_{01,11}^i & f_{01,12}^i & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

The ReLU function replaces all negative entries in $\mathbf{V}^i \mathbf{x}^i$ with 0 and this operation can therefore be implemented locally by replacing the rows of $\mathbf{V}^i$ corresponding to zero entries in this product with the zero vector, producing the matrix $\mathbf{V}'^i$.

Finally, pooling is implemented by carrying out the corresponding operation on the rows of $\mathbf{V}'^i$. For max pooling, only those rows corresponding to the chosen maxima in each pool are copied to $\mathbf{W}^i$, or for average pooling, the rows corresponding to the pooled values are averaged together to form the rows of $\mathbf{W}^i$.

This construction allows us to create a matrix $\mathbf{W}^i$ which locally maps between the input and output planes of a convolutional block, and by multiplying these matrices together, we compute the locally linear map between the input and output of a CNN.

4.5 Interpolation between training samples

The linear mapping between input and output is only valid in a region where all non-linear elements remain in the same state. That means all negatively saturated ReLU units stay negatively saturated, all ReLU units in the linear region remain so, and all max-pooling operations select the same argument. Any investigation of network behaviour must deal with switches between different linear domains. Therefore it is essential to interpolate between neighbouring training samples to understand how different CNNs generalize away from the training samples. This process requires us to track how the linear domains change as we move between those samples. Our initial investigations in this regard revealed two challenges.

- i The max-pool arguments switch rapidly, which requires investigating a large number of linear domains, even for small changes in the input.
- ii Linear interpolation between input samples creates nonsensical interpolated images. This should have been expected, given the highly non-Euclidean nature of the input space.

Two mechanisms can be invoked to address these issues. On the one hand, we have shown that the benefits of CNN depth are equally clear in networks with *average pooling* rather than max pooling, even if the absolute performance is somewhat better with max pooling. Thus, we can address the first issue by modifying the above algorithm to utilize average pooling.

The second proposal is to develop a modified interpolation scheme that accounts for the properties of input space. Let us consider interpolation between two inputs $\mathbf{x}$ and $\mathbf{y}$ and let us restrict our attention to binary images to simplify the calculation. These images differ in d positions and interpolation in n steps requires that these d different values be systematically inverted or flipped. We therefore propose that, on each successive step of interpolation, d/n of the image locations with differing values be changed.

The value of the linear representation now comes into play since we can choose those d/n locations to minimize the change in the output values or logits, thus implementing a suitable form of *interpolation*. To minimize the change in the logits, we want to choose the first interpolated image x_1 to minimize

$$\|\mathbf{W}(\mathbf{x})(\mathbf{x} - \mathbf{x}_1)\| = (\Delta_1^T \mathbf{W}(\mathbf{x})^T)(\mathbf{W}(\mathbf{x})\Delta_1),$$

where Δ_1 is the change vector $\mathbf{x} - \mathbf{x}_1$. Since $\mathbf{W}(\mathbf{x})$ can be calculated with the process described in Section 4.5, this minimization with respect to Δ_1 is a simple matter in principle. In practice, though, it may be necessary to do so by steepest descent to limit computational cost.

We can use this process to generate a sequence of images $\mathbf{x}, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n = \mathbf{y}$, producing similar outputs and interpolating between the original images. As mentioned above, this should give us significant intuition about the invariances representing the network in the neighbourhood of $\mathbf{x}$ and $\mathbf{y}$.

4.6 Results

4.6.1 Binary image interpolation

Our application of the concepts introduced in the previous section begins with interpolating two random binary images from the same class. A binary image in this context was created by assigning each pixel value greater than zero to one and leaving the zero pixels unchanged. Figure 4.2 shows a pair of samples from each class in the MNIST dataset, with standard images to the right and binary versions to the left. The algorithm in Section 4.5 calls for a simple pixel-by-pixel comparison between pairs of images to determine the modified pixels. These pixels are used to calculate the amount of change for each successive interpolation step. For example, the first and second binary samples from class zero in Figure 4.2 have 106 mismatched pixels. If we opt to interpolate from first to second sample in ten steps, we can change the following number of pixels in sequence: [11, 11, 11, 11, 11, 11, 10, 10, 10, 10]. Meaning, 11 pixels should have changed after the first interpolation step. the process continues until there are 10 pixels left to change in the tenth (final) interpolation step. The choice of which bit to flip was based on which change affected the output logits the least. This process was conducted for CNNs of different depths and a single-layer MLP network of adequate capacity, and typical examples of the results are shown from Figure 4.3 to Figure 4.12.

The interpolation plots show that the deeper the network, the smoother the interpolated images compared to shallower ones. This means the first bits to change seem less random and more precise, and the digits are kept intact for deeper networks. Subsequent interpolations also seem systematic and visually sensible for deeper networks. This is most noticeable in cases where the two anchor images are substantially different from one another, as in Figure 4.11: all the networks create significantly distorted interpolated images, but the interpolations by the 4-layer network are (for example) more image-like than those of the 1-layer network or MLP.

There is a particularly significant difference between networks with 0, 1 and 2 convolu-

tional layers, on the one hand, and those with 3 and 4 layers – networks with 4 layers behave much the same as those with 3 layers. This was also observed in our validation accuracy results and may result from the rather simple MNIST task; we expect the same trend to continue to deeper networks with higher-resolution images and more complex classification tasks. In addition, as we add more convolutional layers, the networks seem to generalize more appropriately, which is indicated by the structure of the digits between interpolation steps. This reasoning comes from the fact that all these networks of different depths were trained close to 0% error rate and had similar capacities. From these observations, it would be reasonable to suggest that something inherent or natural in the deeper CNNs leads to this natural-looking interpolation. Our concluding discussion explores this capability of deeper networks in more detail.

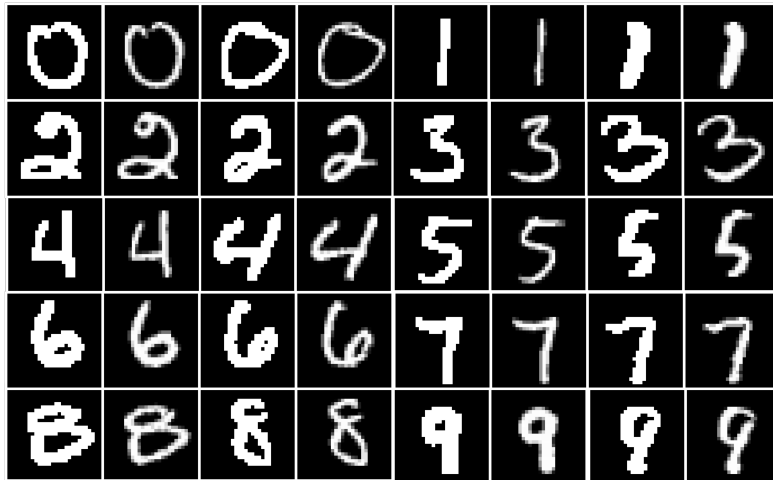


Figure 4.2: MNIST standard and binary samples. Samples on the left represent binary images with only 0s and 1s. Note that all pixel values greater than 0 were assigned to 1.

Based on our intuition above, a CNN with more convolutional layers has more reasonable interpolation. A reasonable image (if one interpolates between two intra-class images, for example, 4s, then the intermediate images must be 4-like) is fairly smooth and intact, as shown from Figure 4.3 to Figure 4.12. Similar reasoning should apply for inter-class images or images from different classes, as shown with the interpolation from class 4 to class 8 in Figure 4.13 and in A.2. Each intermediate image must be smooth to generalize well for all the randomly selected samples. This implies that the deeper the network (more hidden layers), the smoother and more intact the intermediate images. Intermediate images in

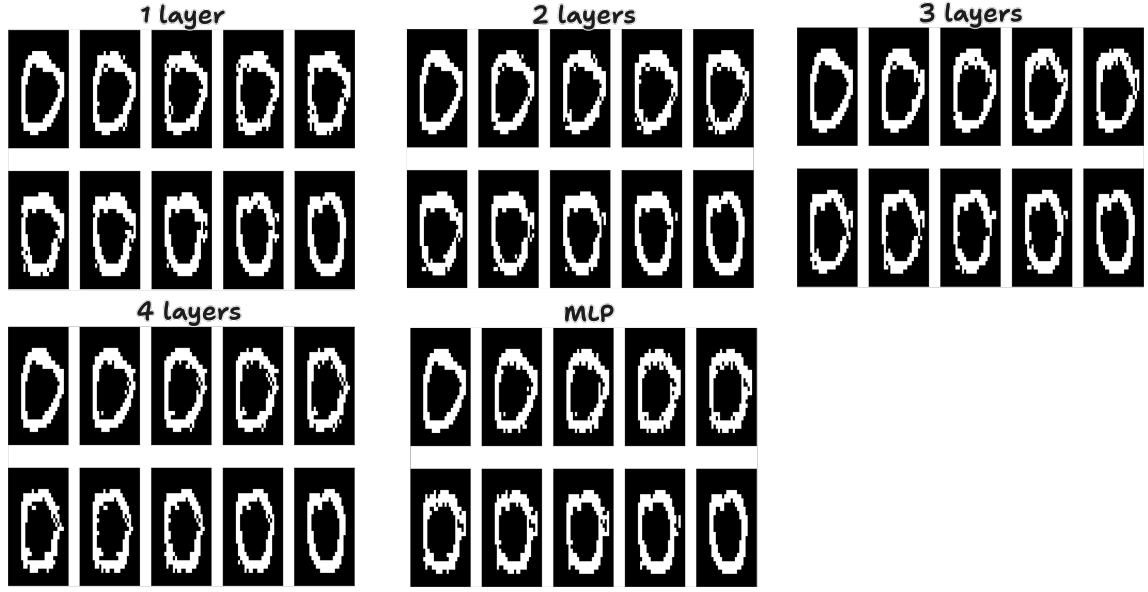


Figure 4.3: MNIST class 0 intra-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.

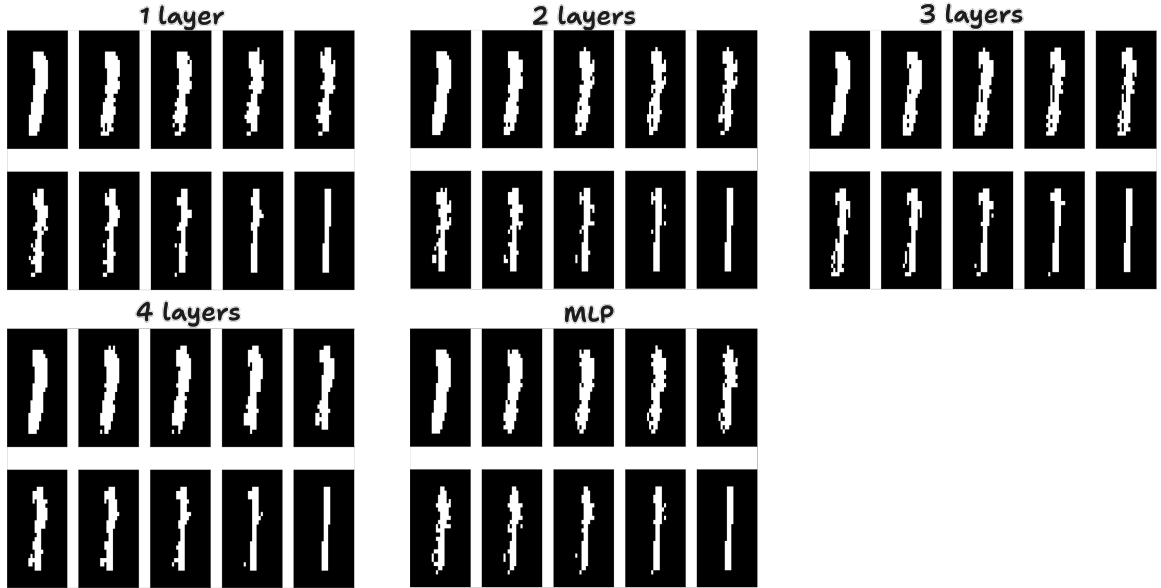


Figure 4.4: MNIST class 1 intra-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.

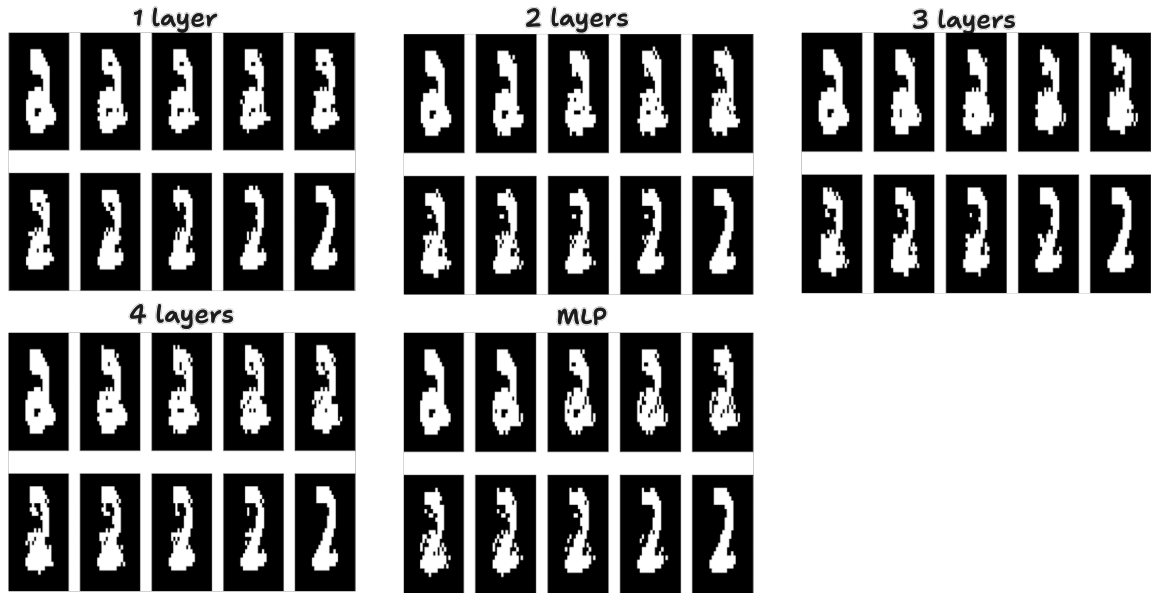


Figure 4.5: MNIST class 2 intra-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.

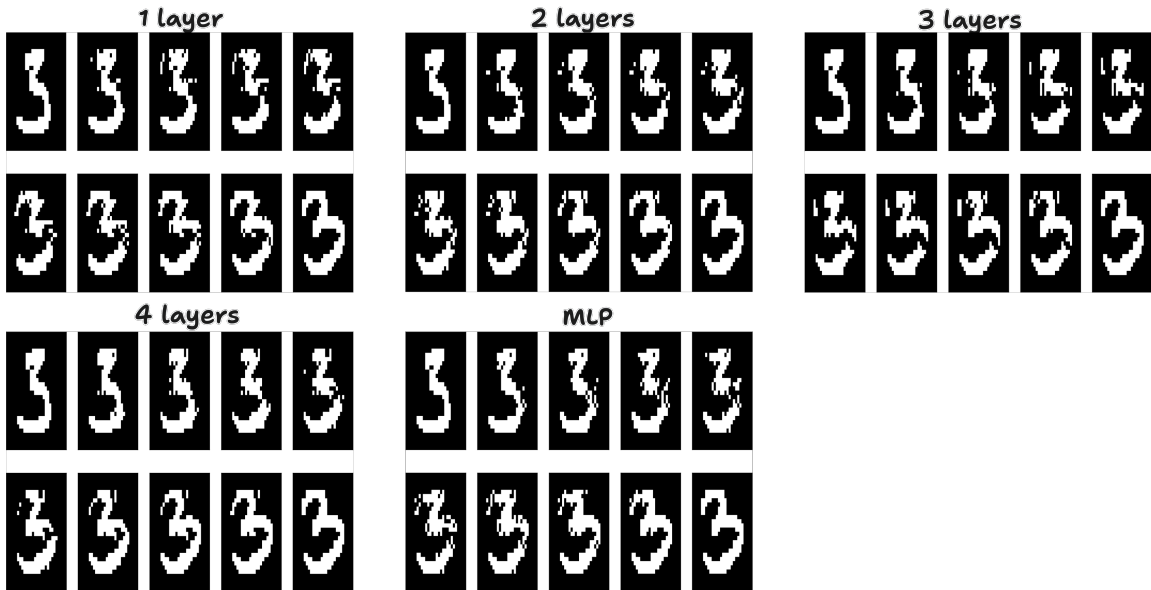


Figure 4.6: MNIST class 3 intra-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.

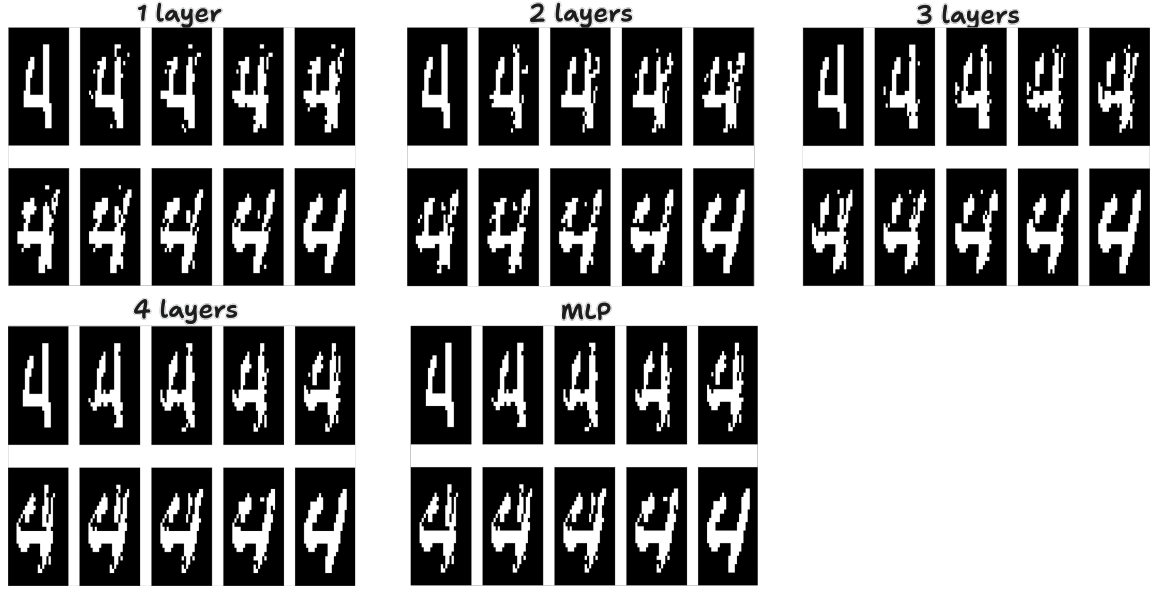


Figure 4.7: MNIST class 4 intra-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.

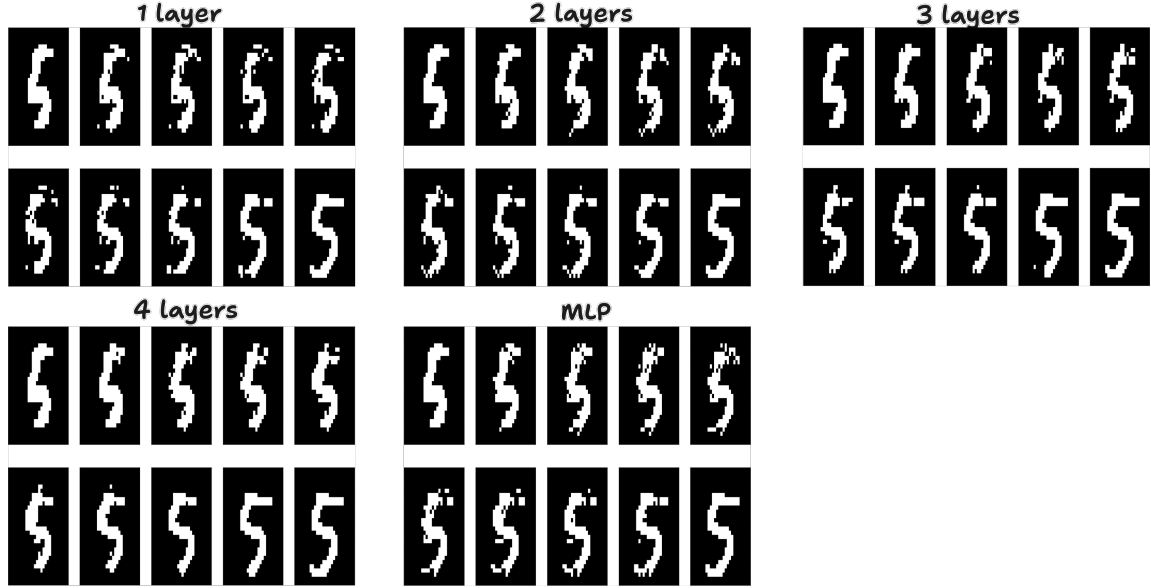


Figure 4.8: MNIST class 5 intra-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.

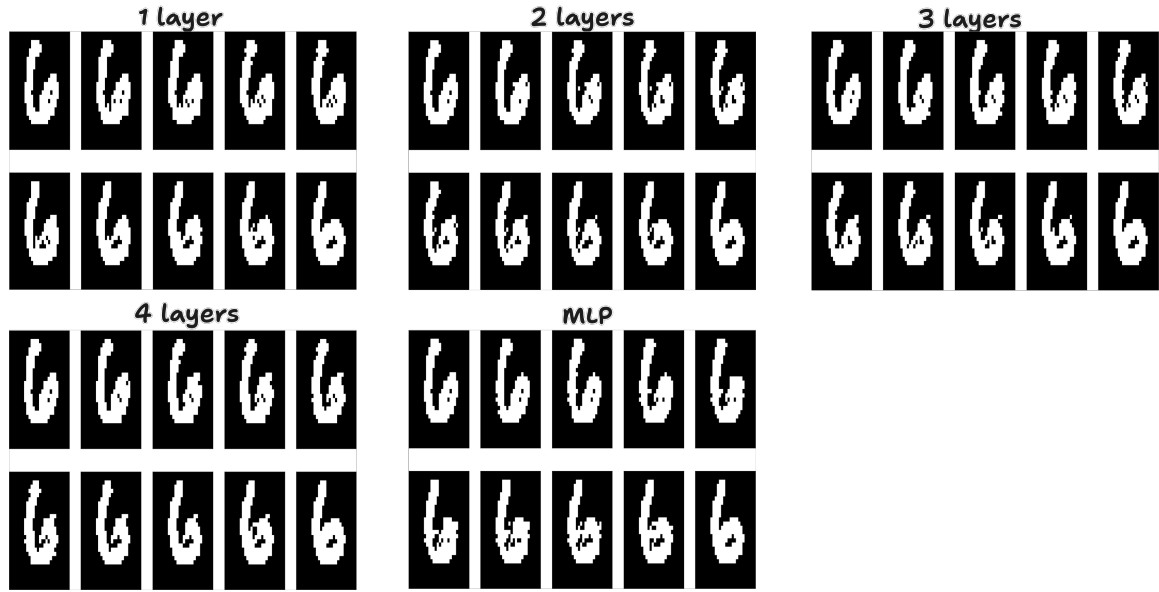


Figure 4.9: MNIST class 6 intra-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.

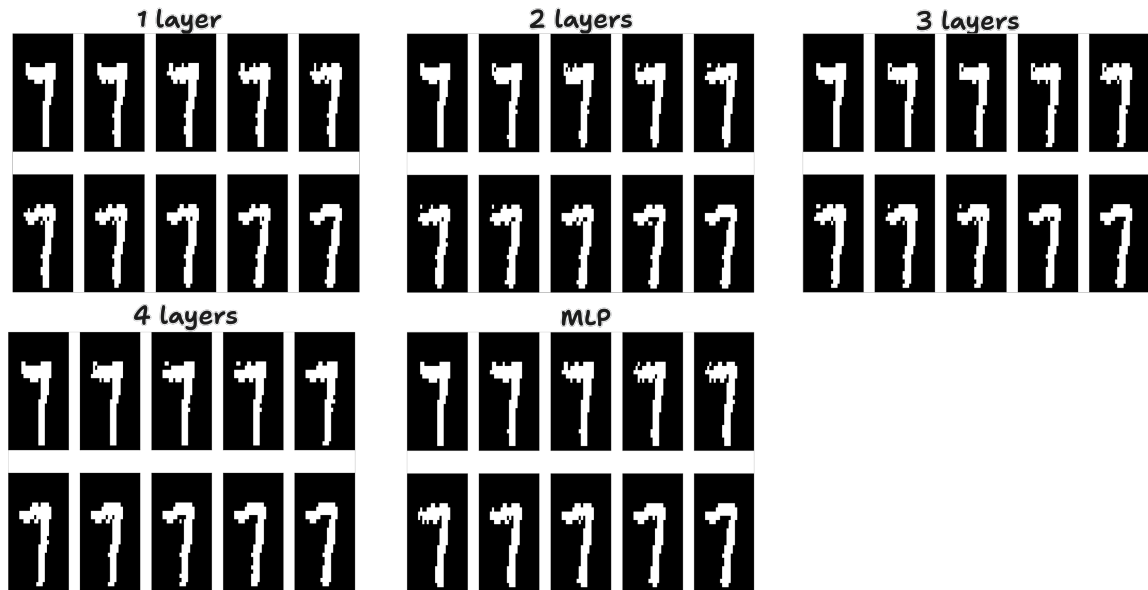


Figure 4.10: MNIST class 7 intra-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.

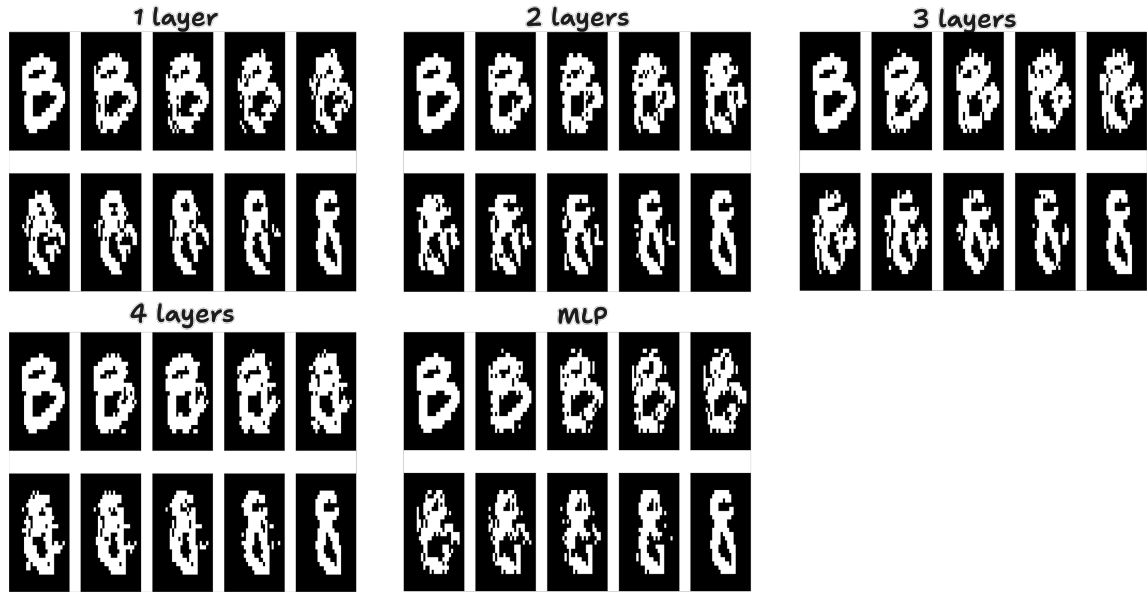


Figure 4.11: MNIST class 8 intra-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.

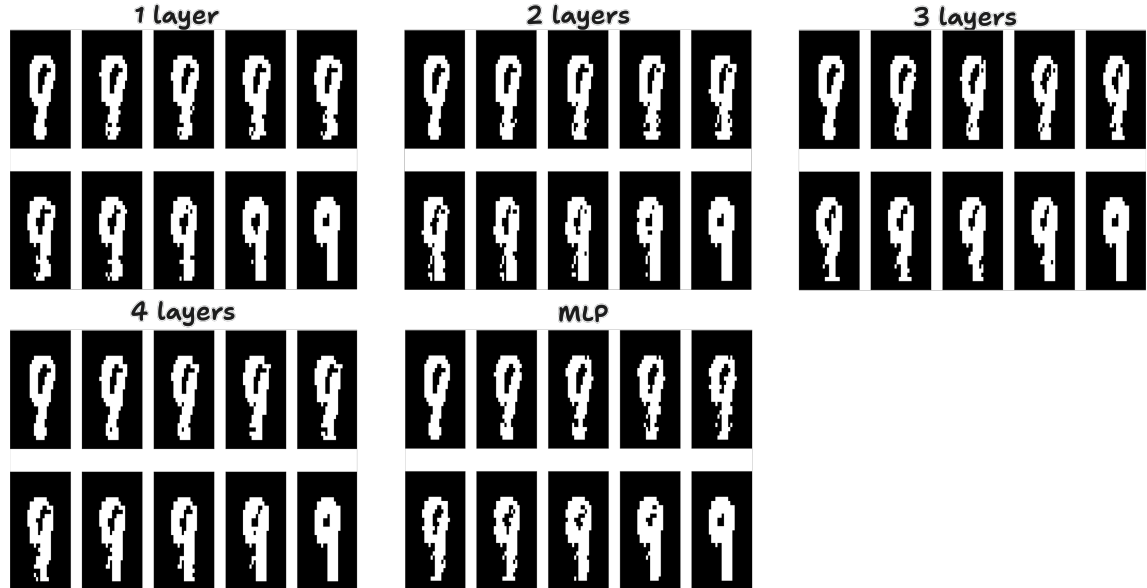


Figure 4.12: MNIST class 9 intra-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.

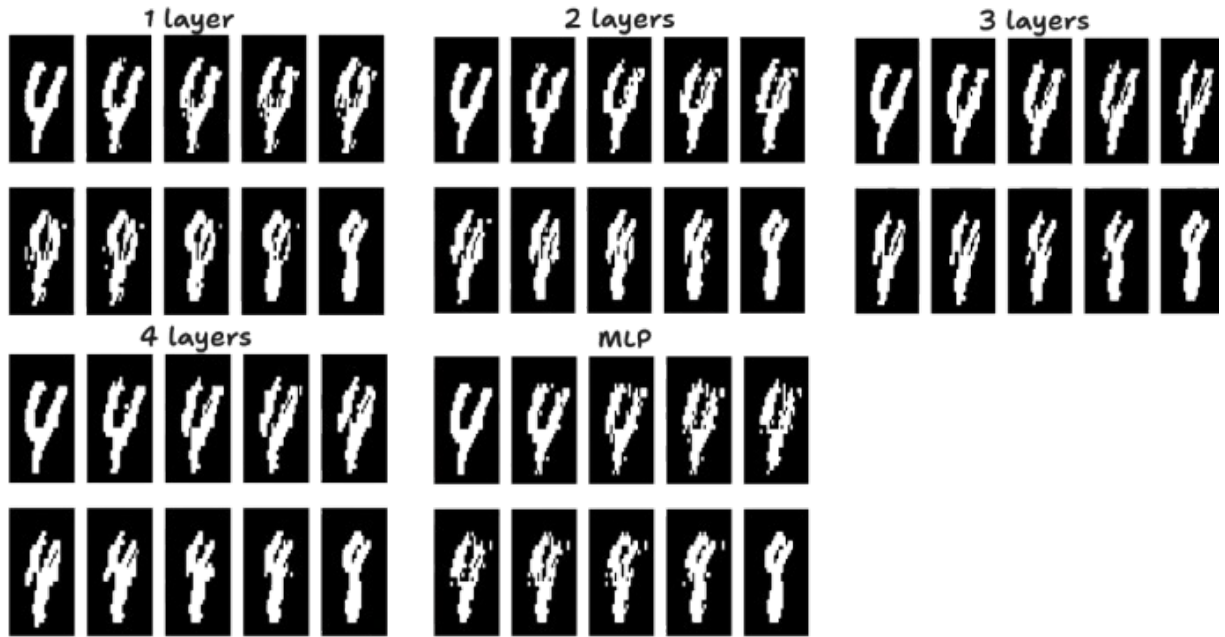


Figure 4.13: MNIST classes 4 and 8 inter-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.

deeper networks, therefore, have more low frequency components than shallower networks, and shallower networks have more low frequency components, isolated pixels and more random-looking pixels. We will now introduce numerical measures aimed at capturing this intuition.

4.6.2 2D FFT binary images

To measure the actual difference between the intermediate images for the different layers, we do not just consider the number of bits but also the placement of the bits. We used the 2-dimensional Fast Fourier Transform (2D Fast Fourier Transform (FFT)) to analyze the frequency spectrum of the 2D signal or matrix. This will assist in quantifying the frequency differences between the intermediate binary images. From Figure 4.5, each interpolated image was processed as follows:

- Transform the image using the 2D FFT

-
- Shift the frequency spectrum to center zero frequency components
 - Apply the absolute value to the FFT

This allowed us to visualize the image symmetry, focus on only the magnitude or intensity of the signal at each frequency point and disregard the phase. Results are shown in Figure 4.14 for each interpolation stage for four network depths and the MLP network. The images were still not distinguishable between interpolations and required further analysis. Due to the complex conjugates resulting from the coefficients of positive and negative frequencies, each real image produced a two-fold symmetry with a low frequency component concentration around 0 Hertz [98], as shown in Figure 4.15a. So, we needed to determine the separation between high and low frequency components for each image, as shown in Figure 4.15d, to ensure clear differences between the various depths. In addition, the symmetries needed to be carefully summed to avoid duplication, as indicated in Figure 4.15b.

We measured the radial or Euclidean distance from the centred frequency component of Figure 4.15c to a selected radius. For example, if the radius $\mathbf{R} = 4$, we summed all the frequency components, including the DC value within the mesh grid (within the black border) shown in Figure 4.15d. Therefore, we present the measure of the difference between interpolations for various depths as the average of the normalized sum of the frequency components within the mesh grid. Multiple iterations were used for each interpolation stage to generate the average sum. A clear distinction between classes may already be visible within the first 4 iterations. However, we elected to increase our iterations to 10. This ensured that we generated stable outputs due to the varying behaviour of the different class samples. Figure 4.16 shows the 2D FFT sum of the frequency components within the radial distance for the MNIST dataset, as described above. The left and right extremes represent the average sum for the randomly selected non-interpolated samples for a given class. Since the interpolated images are not real digits but the network’s synthesis of the digit, the perfect prior would generate a plot that directly connects to the two extremes. Therefore, the networks with the best prior provide values closest to the straight line between the two extremes. The figure also indicates that the MLP and single convolution

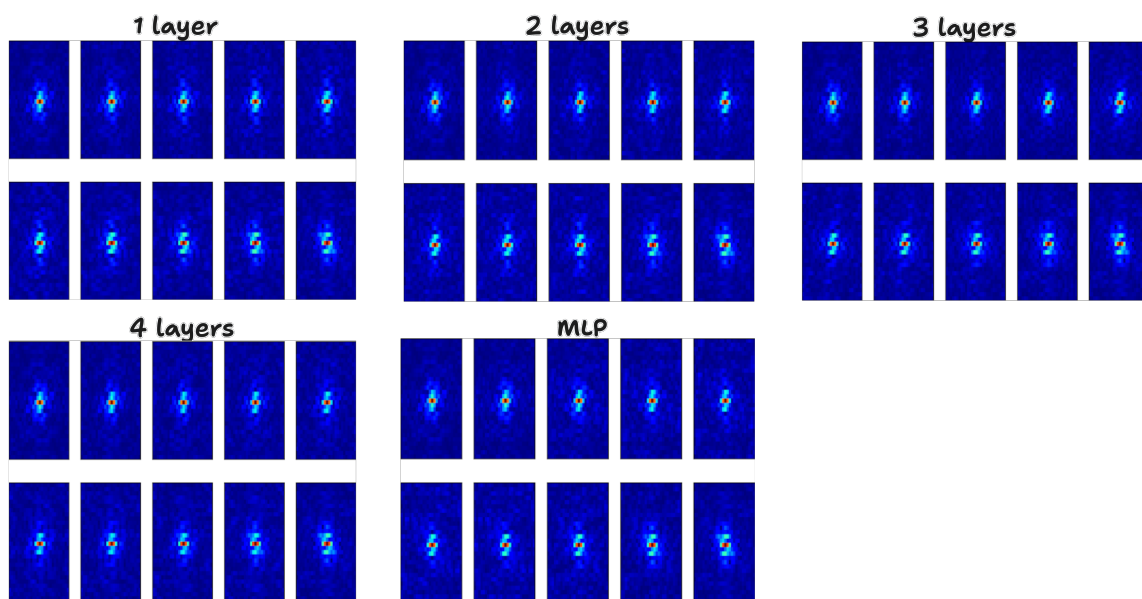


Figure 4.14: MNIST class 2 sample interpolation with 2D FFT for the different CNN depths and an MLP network. Each subsequent interpolation stage was transformed from Figure 4.5.

layer networks provide the worst priors. Conversely, networks with more hidden layers generate better priors and improve as network depth increases. Similar patterns can be observed for a more challenging FMNIST dataset and across classes, depicted by Figure 4.17 and Figure 4.18 respectively.

4.6.3 Bit change summations

One can imagine various sensible metrics to measure the difference between the intermediate images for the layers shown in Figure 4.14. These metrics should consider both the number of active pixels and the placement of those pixels. One obvious measure is the 2D FFT, which analyzes the frequency spectrum of the 2D signal or matrix. However,

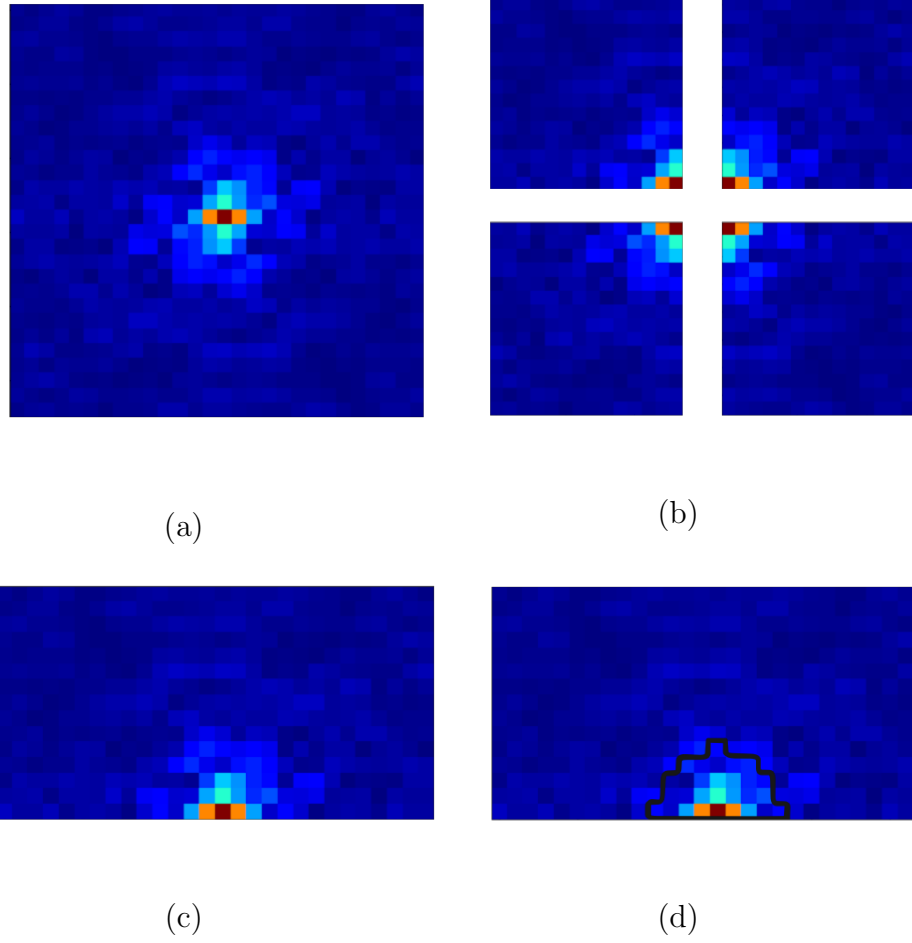


Figure 4.15: An intermediate binary shifted 2D FFT plot for a) single symmetric interpolated image for a class 2 MNIST sample. b) clear symmetric illustration of the even and odd quadrants with centred frequencies. c) half 2D image sample for frequency calculations without symmetry redundancy. d) half 2D sample with $\mathbf{R} = 4$ mesh grid for frequency summation.

FFT-based smoothness measures can be problematic because they suffer from anisotropy and behave differently in coordinate directions than in rotated directions. In addition, the frequency cut-off that defines low frequency components is a free parameter with appropriate values likely to be problem-specific. Therefore, we explored a simple alternative smoothness measure that was easier to substantiate and not dependent on fine-tuning.

Based on the differences observed in Figure 4.14, we wish to define a measure that identifies images that do not contain “unnatural” pixels. To this end, we determine whether every

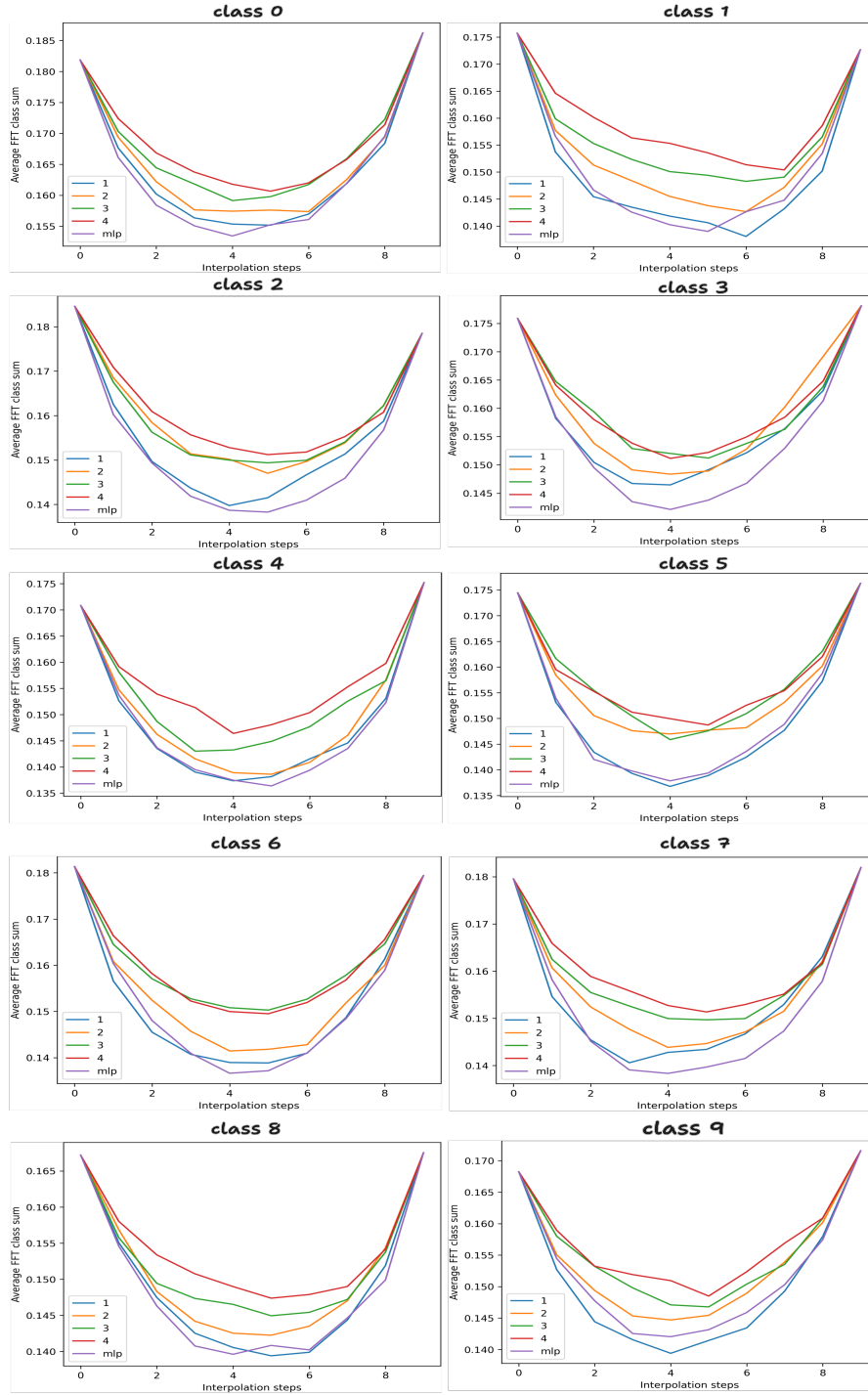


Figure 4.16: MNIST intra-class average 2D FFT sums for 4 CNNs and MLP network. The numbers within the legend represent the depth of each CNN and the MLP represents a single-layer, large-capacity network. Each interpolation point is computed using an average of 10 2D FFT sums.

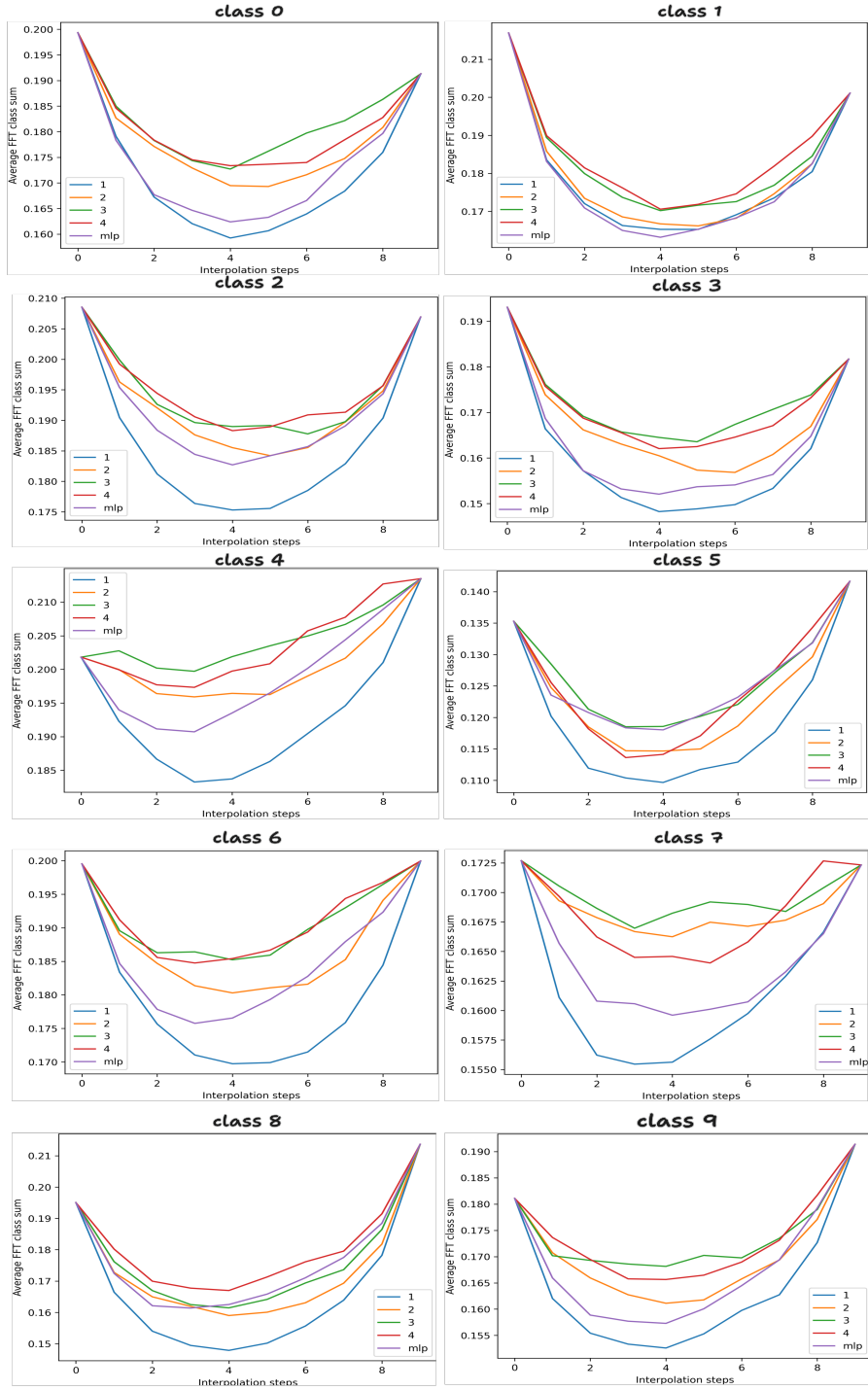


Figure 4.17: FMNIST intra-class average 2D FFT sums for 4 CNNs and MLP network. The numbers within the legend represent the depth of each CNN and the MLP represents a single-layer, large-capacity network. Each interpolation point is computed by averaging over 10 2D FFT sums.

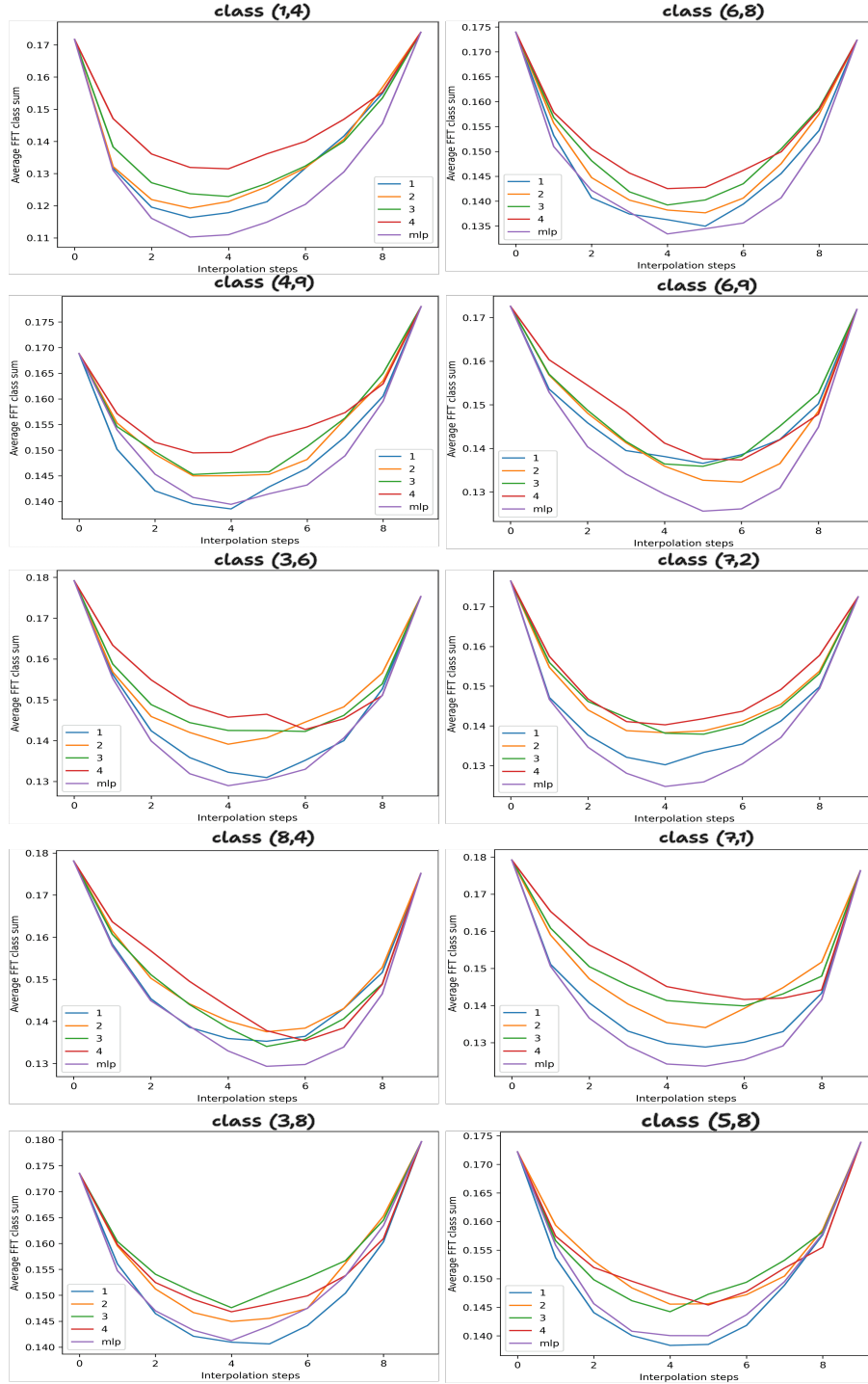


Figure 4.18: MNIST inter-class average 2D FFT sums for 4 CNNs and MLP network. The numbers within the legend represent the depth of each CNN and the MLP represents a single-layer, large-capacity network. Subplot titles indicate the different interpolated classes. Each interpolation point is computed by averaging over 10 2D FFT sums.

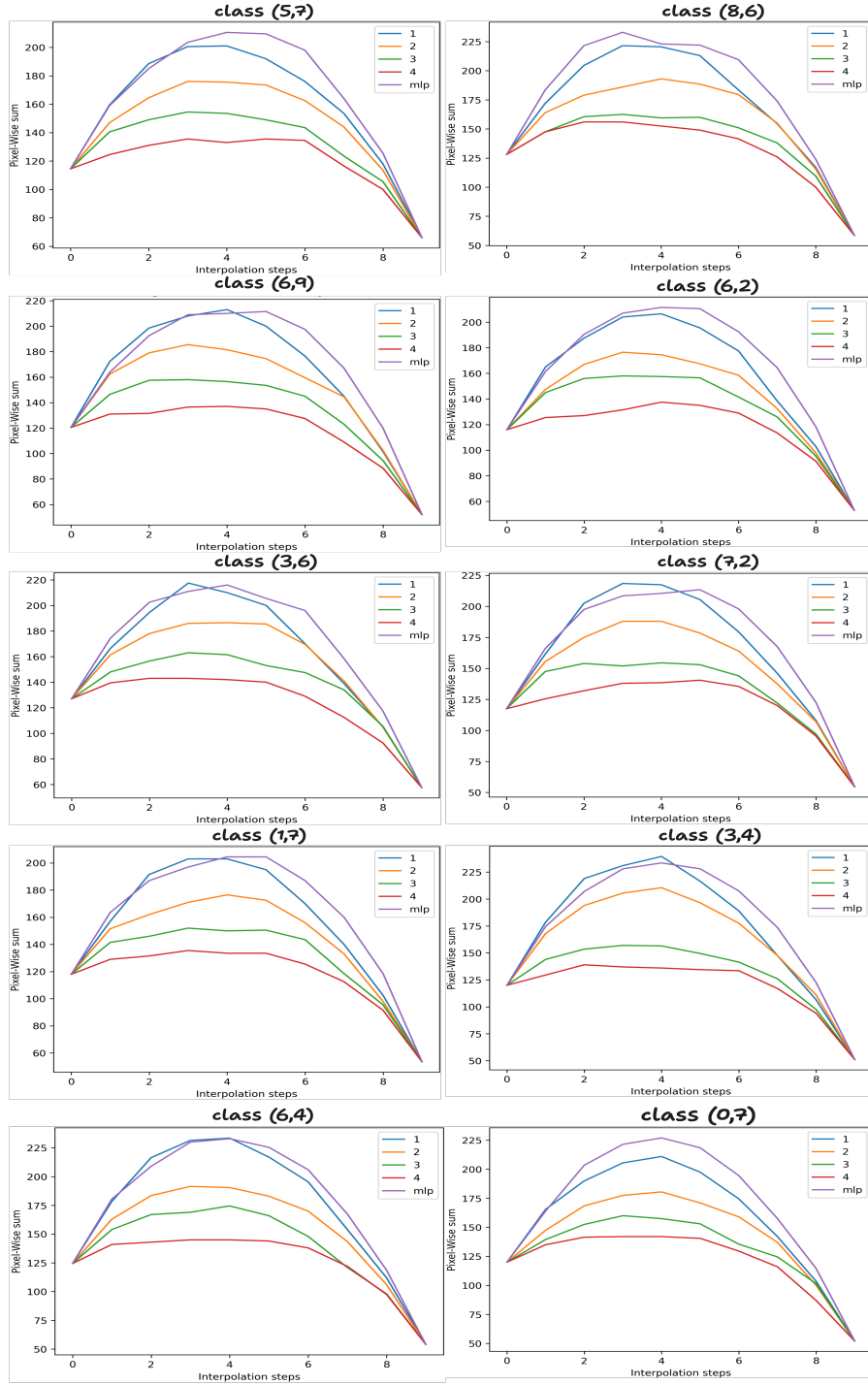


Figure 4.19: MNIST inter-class bit count for 4 CNNs and MLP network. The numbers within the legend represent the depth of each CNN and the MLP represents a single-layer, large-capacity network. Subplot titles indicate the different interpolated classes. Each interpolation point is computed by adding vertical and horizontal bit changes.

pixel is surrounded by other pixels with similar characteristics since a smooth binary image is one where most black pixels are surrounded by other black pixels and similarly for white pixels. Moreover, a real image with a good prior (one that generalized well) should be more connected and continuous, while a bad prior (one that did not generalize well) should introduce random-looking bits and pieces. So, to compute the smoothness measure, the binary image was scanned pixel by pixel from left to right to determine how many of each pixel’s horizontal neighbours differed from the current pixel. The scanning process counted the number of times white pixels changed to black and black pixels to white. The exact process was repeated by scanning the pixels from top to bottom and counting the vertical changes. Both counts were added to give the number of un-surrounded edges a binary image had. This was performed for each interpolated binary image, for N iterations, for different CNN depths and a single-layer large-capacity MLP network.

Figure 4.19 and Figure 4.20 show the bit flip counts for various CNN depths and the MLP network for the MNIST dataset. The extreme points represent the start and end average bit change counts for randomly selected binary images. A perfect interpolation would result in a straight line between the extreme points. Based on the observation, the MLP network seems to have the most bit switches on average for every class and thus generates intermediate images that are the least image-like. CNNs, on the other hand, performed better on average for all the classes and showed some improvement as more hidden layers were introduced. A clear performance distinction exists between a CNN with 1 hidden layer and 2 or more hidden layers. In addition, a network with 3 hidden layers seems to perform very similarly to the one with 4 hidden layers, although the latter performs the best on average. These relationships agree well with the model performance results in Section 3.2. Similar observations were made for FMNIST intra-class and inter-class dataset, as shown in Figure 4.21 and Figure 4.22 respectively. In this case, the superiority of the deeper networks is more substantial, as evidenced by the lower average bit change counts for the interpolated binary samples.

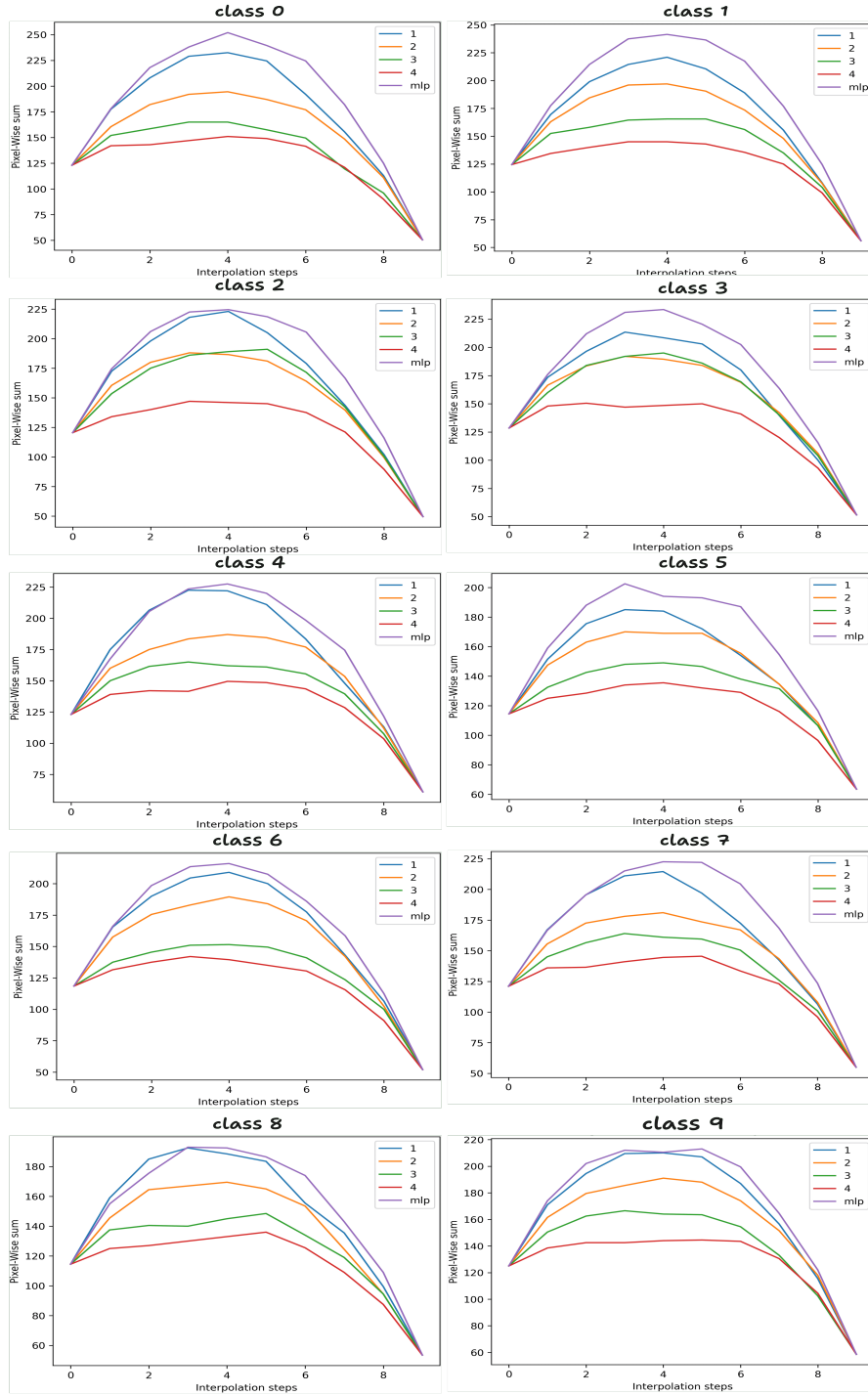


Figure 4.20: MNIST intra-class bit counts for 4 CNNs and MLP network. The numbers within the legend represent the depth of each CNN and the MLP represents a single-layer, large-capacity network. Each interpolation point is computed by adding vertical and horizontal bit changes.

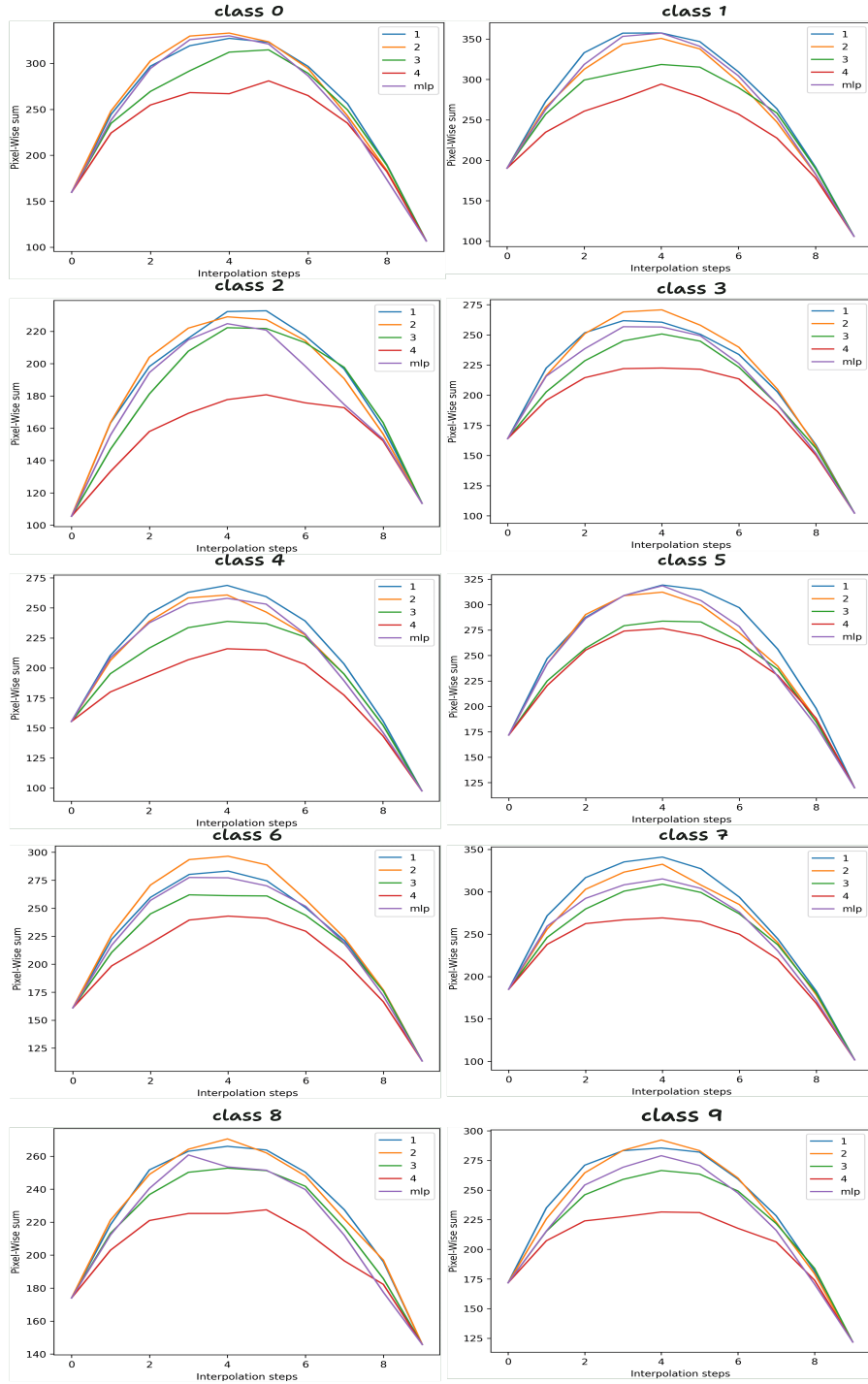


Figure 4.21: FMNIST intra-class bit counts for 4 CNNs and MLP network. The numbers within the legend represent the depth of each CNN and the MLP represents a single-layer, large-capacity network. Each interpolation point is computed by adding vertical and horizontal bit changes.

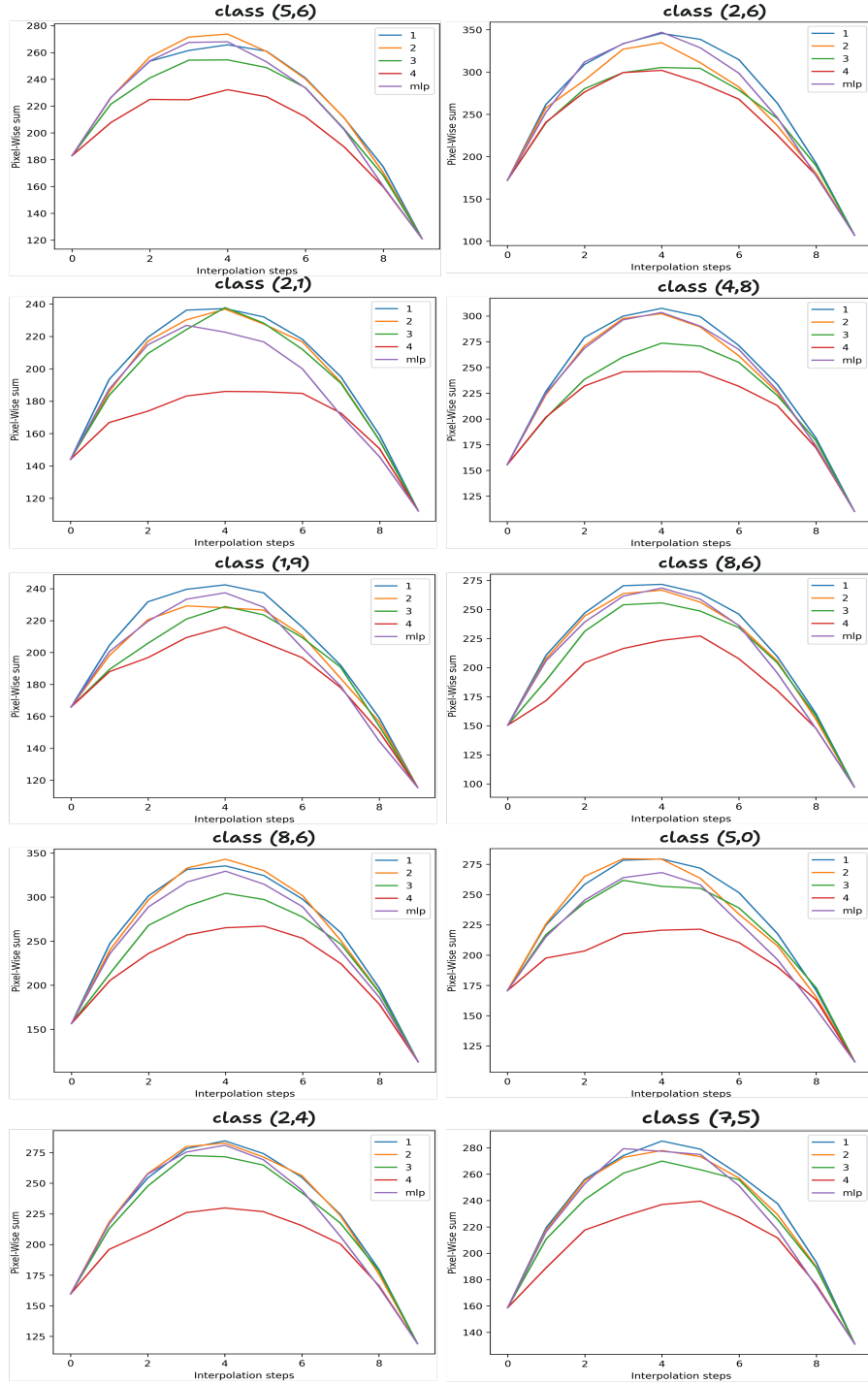


Figure 4.22: FMNIST inter-class bit count for 4 CNNs and MLP network. The numbers within the legend represent the depth of each CNN and the MLP represents a single-layer, large-capacity network. Subplot titles indicate the different interpolated classes. Each interpolation point is computed by adding vertical and horizontal bit changes.

4.7 Discussion

This chapter presented a framework that viewed classifiers as converters or maps between the feature space and output space to understand the generalization capabilities of CNNs. We developed a precise theoretical expression to describe the piecewise linear representation of a convolutional layer. In addition, we investigated the properties of such converters using linear interpolation between in-class training samples, applying the linear representation to probe their properties.

Our experiments showed that the maps for CNNs are biased to be increasingly smooth with greater depth, according to two different metrics. Since such smoothness is a characteristic of real images, deeper CNNs tend to outperform shallow CNNs and MLPs. By mapping pixels into feature space, Zeiler et al. [92] observed the hierarchical nature of CNN layers where early layers respond to simple features and later ones to more complex features. These observations may represent the mechanism whereby deeper networks have a more image-like bias since they can prefer ever-larger patches of realistically smooth input structure. For the purposes of this conclusion, either of the two definitions of smoothness introduced above, FFT-based and alternation-based, would be sufficient. More sophisticated measures of image-like quality can certainly be developed to better characterize the advantages of deeper CNNs.

Wolpert et al. [99] introduced the Extended Bayesian Framework (EBF) to study the inductive bias of learning algorithms. There are reasons to be sceptical about the most celebrated result of that framework, namely the "No Free Lunch" theorem; these reasons are explored in [100]. We nevertheless believe that the EBF contains the appropriate theoretical elements to explain our results in a general setting. In particular, the EBF extends conventional probability theory by treating the hypothesis that is output by a learning algorithm as a random variable h . A classifier is represented by a probabilistic relationship between the training data d and the function f that represents the input-output relationships of the classifier. The generalization error is expressed by considering the dependencies of the (off-training set) error on all three of these variables (h , f , and

d). The correspondence between the prior distributions of h and f is therefore an analytic expression for the inductive bias represented by a given learning algorithm.

Chapter 5

Conclusion

In this chapter, we provide an interpretation of our results, summarise the findings and further expand on the most meaningful contributions of our research. In addition, we provide guidelines for envisioned future work.

5.1 Overview

This thesis presented scientific approaches towards understanding the concept of generalization in standard deep CNNs. With this overarching goal in mind, we explored quantifiable methods to demystify the underlying contrasts between deep CNNs and MLPs. Considering other approaches from the literature, we believe that our framework to view classifiers as converters or mappings between input and output space contributed to the fundamental understanding in ML and Deep Learning (DL). We discuss the significance of our contributions and key findings in Section 5.2. Section 5.3 proposes additional and alternative avenues for future research. Section 5.4 recaps the findings and final remarks.

5.2 Key findings

5.2.1 Investigating importance of depth in CNNs

To effectively answer our research questions and set the basis for analysis and further insights, we established simple MLP networks and multi-layered CNN architectures to observe the systematic effects of greater depth. We consolidate our main observations as follows:

- Increased depth proved advantageous in CNN architectures. Networks with additional layers showed a methodical decrease in validation error performance across all datasets. This pattern was consistent for lower and higher capacity networks. MLP architectures did not derive the same benefits from increased depth. It is worth noting that deeper MLP networks may be utilised to streamline training due to their reduced trainable parameters. Still, from the perspective of generalization, additional layers are not particularly useful in MLP architectures (Section 3.4).
- An increase in capacity or number of neurons showed significant improvements in the validation results of the CNN networks. Comparatively, MLP networks did not significantly improve the validation performance across all datasets.
- An increase in trainable parameters did not degrade the validation error rate for lower and higher capacity networks. On the contrary, CNN networks with more trainable parameters showed superior validation performance across all datasets.

These observations contrasted with the SLT and its key concept of the bias-variance tradeoff, which implies that over-parameterized, high-dimensional and high-complexity networks should not perform well on unseen data.

5.2.2 Investigating the piecewise linear model

Notwithstanding the empirical evidence, there was still no conclusive explanation of why depth in CNNs was preferred and superior. To further appreciate these generalization capabilities, we took a different approach to SLT by proposing the concept of classifiers as converters. Our observations are summarised as follows:

- Interpolation between two binary neighbouring training samples on networks with different numbers of convolutional layers suggests that networks with more layers systematically prefer smoother, more image-like interpolations. This was corroborated by the compactness of intermediate images resulting from deeper networks (Section 4.6).
- The frequency analysis revealed that binary samples interpolated by deeper CNNs are more compact and smoother. The sum of the frequency components within the radial threshold was significantly less in deeper networks, which implied fewer frequency changes. This observation was consistent for interclass and intraclass samples.
- Simple bit counts, which served as an alternative measure of smoothness, demonstrated that deeper CNNs produced interpolated samples where pixels with similar characteristics surrounded each other. Deeper CNNs generally produced lower horizontal and vertical bit changes for each interpolated sample across all datasets.
- MLP networks generally performed worse than all the CNN networks in all measures, consistent with the model output results.

Our observations also showed that the MLP networks had higher average bit change for all classes and, as a result, produced interpolation images that were less compact and less smooth. CNNs, on the contrary, performed better on average by producing smoother and more compact interpolations. This systematic improvement was observed as more hidden layers were introduced for all classes.

These results strongly suggest that deeper CNNs are able to generalize more successfully because they are biased to prefer greater smoothness or less roughness in the input space. This agrees with the finding that deeper layers tend to respond to increasingly compound image structures [92] and stands in contrast to the idea that shift invariance is an essential capability in this regard. In fact, it has long been known [101] that CNN-type architectures cannot be strictly shift invariant.

The classification results of Section 3.4 express a number of essential characteristics. Firstly, there is a systematic improvement in generalization when the fit between the target function and the hypothesis class improves, and this improvement can be understood in terms of the characteristics of the target function. Secondly, the number of parameters does not play an essential role in the observed trends. It may, however, be possible to demonstrate the bias-variance trade-off in these examples under certain circumstances, e.g. with no early stopping and substantial overparametrisation of the classifiers. Even so, this would be a second-order effect and typically much smaller than the differences observed, especially in the CIFAR-10 dataset.

5.3 Future work

While the work completed throughout this study was confined to addressing the formulated research questions, empirical results and observations opened other avenues for further research opportunities.

- The EBF provides a promising way to consider these generalisation patterns since it focuses on an abstract but deep relationship between the target function and the hypothesis class (e.g., the neural network’s architecture employed). We believe this theoretical framework has great promise in research on generalization not only for architectures such as those we have discussed but also for more modern architectures such as Transformers and more classical architectures such as decision trees. For this potential to be realized, new tools and additional practical insights are required. To

progress with CNNs, we employed intuitive metrics derived from our understanding of their operation. It would be very useful if systematic tools could be developed to estimate the fit between appropriately defined target classes and different hypothesis classes.

- Implementation of the piecewise linear model required using a Toeplitz-like matrix to represent an all-encompassing weight vector. This approach is practical for lower-resolution single-channel datasets such as MNIST and FMNIST. However, this approach is impractical for processing higher-resolution multi-channel datasets due to the size of the input feature representations. Further investigation into suitable optimisation approaches or alternative approximations would be useful in making the model more efficient.
- The smoothness metric introduced in Section 1 shows systematic relationships with generalization accuracy across many tasks and classes. It would therefore be useful to investigate its properties more carefully. For example, it would be interesting to see whether those classes with the best apparent smoothness for a model (according to our metric) are also relatively accurately recognized by that model. Also, what is the range of smoothness values for data in our sets, and are outliers by this measure relatively hard to recognise? Findings on these matters would not affect our main conclusions but would lead to a better understanding of the proposed metric.

5.4 Final remarks

Our quest to understand generalization in DNNs started with the observation that additional hidden layers are valuable in CNNs but not so much (if at all) in MLPs. Through a detailed analysis, we were able to show how greater depth in the former network type corresponds with a bias in favour of smoother images, which is not true of the latter type. Beyond resolving our main research question, this insight represents (in our opinion) a novel and concrete actualization of what an inductive bias may be. Earlier work had generally equated “inductive bias” with the preference for simplicity, also known as Oc-

cam’s razor. Our findings show that other instantiations of that bias are more relevant in CNNs. We imagine that this expanded view of inductive bias may be the most enduring contribution to our work.

Bibliography

- [1] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, 1943.
- [2] M. Minsky and S. Papert, “An introduction to computational geometry,” *Cambridge tiass., HIT*, vol. 479, no. 480, p. 104, 1969.
- [3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *nature*, vol. 323, no. 6088, pp. 533–536, 1986.
- [4] A. Krizhevsky, G. Hinton, *et al.*, “Learning multiple layers of features from tiny images,” 2009.
- [5] J. De Villiers and E. Barnard, “Backpropagation neural nets with one and two hidden layers,” *IEEE transactions on neural networks*, vol. 4, no. 1, pp. 136–141, 1993.
- [6] A. Vaswani, “Attention is all you need,” *arXiv preprint arXiv:1706.03762*, 2017.
- [7] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, “A neural probabilistic language model,” *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.
- [8] C. Wang, M. Li, and A. J. Smola, “Language models with transformers,” *arXiv preprint arXiv:1904.09408*, 2019.
- [9] L. Rabiner and B.-H. Juang, *Fundamentals of speech recognition*. Prentice-Hall, Inc., 1993.

-
- [10] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, “Convolutional neural networks for speech recognition,” *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014.
 - [11] D. H. Ballard and C. M. Brown, *Computer vision*. Prentice Hall Professional Technical Reference, 1982.
 - [12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, vol. 25, 2012.
 - [13] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, “Understanding deep learning (still) requires rethinking generalization,” *Communications of the ACM*, vol. 64, no. 3, pp. 107–115, 2021.
 - [14] S. Geman, E. Bienenstock, and R. Doursat, “Neural networks and the bias/variance dilemma,” *Neural computation*, vol. 4, no. 1, pp. 1–58, 1992.
 - [15] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
 - [16] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
 - [17] S. Fortmann-Roe, “Understanding the bias-variance tradeoff. i: Url: [Http://scott.fortmann-roe.com/docs/](http://scott.fortmann-roe.com/docs/),” *BiasVariance.html (hämtad 2017-09-29)*, 2012.
 - [18] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, “Knn model-based approach in classification,” in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings*, Springer, 2003, pp. 986–996.
 - [19] H. Takeda, S. Farsiu, and P. Milanfar, “Kernel regression for image processing and reconstruction,” *IEEE Transactions on image processing*, vol. 16, no. 2, pp. 349–366, 2007.

-
- [20] B. Neyshabur, R. Tomioka, and N. Srebro, “In search of the real inductive bias: On the role of implicit regularization in deep learning,” *arXiv preprint arXiv:1412.6614*, 2014.
- [21] R. Novak, Y. Bahri, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein, “Sensitivity and generalization in neural networks: An empirical study,” *arXiv preprint arXiv:1802.08760*, 2018.
- [22] M. Belkin, D. Hsu, S. Ma, and S. Mandal, “Reconciling modern machine-learning practice and the classical bias–variance trade-off,” *Proceedings of the National Academy of Sciences*, vol. 116, no. 32, pp. 15 849–15 854, 2019.
- [23] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro, “Exploring generalization in deep learning,” *Advances in neural information processing systems*, vol. 30, 2017.
- [24] B. Neal, S. Mittal, A. Baratin, *et al.*, “A modern take on the bias-variance tradeoff in neural networks,” *arXiv preprint arXiv:1810.08591*, 2018.
- [25] Z. Yang, Y. Yu, C. You, J. Steinhardt, and Y. Ma, “Rethinking bias-variance trade-off for generalization of neural networks,” in *International Conference on Machine Learning*, PMLR, 2020, pp. 10 767–10 777.
- [26] V. Quétu and E. Tartaglione, “Dodging the double descent in deep neural networks,” in *2023 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2023, pp. 1625–1629.
- [27] S. d’Ascoli, L. Sagun, and G. Biroli, “Triple descent and the two kinds of overfitting: Where & why do they appear?” *Advances in Neural Information Processing Systems*, vol. 33, pp. 3058–3069, 2020.
- [28] R. Schaeffer, M. Khona, Z. Robertson, *et al.*, “Double descent demystified: Identifying, interpreting & ablating the sources of a deep learning puzzle,” *arXiv preprint arXiv:2303.14151*, 2023.
- [29] C. Y. Liu and J. Flanigan, “Understanding the role of optimization in double descent,” in *OPT 2023: Optimization for Machine Learning*, 2023.
-

-
- [30] V. N. Vapnik, “An overview of statistical learning theory,” *IEEE transactions on neural networks*, vol. 10, no. 5, pp. 988–999, 1999.
- [31] S. Hochreiter and J. Schmidhuber, “Flat minima,” *Neural computation*, vol. 9, no. 1, pp. 1–42, 1997.
- [32] C. S. Wallace and D. M. Boulton, “An information measure for classification,” *The Computer Journal*, vol. 11, no. 2, pp. 185–194, 1968.
- [33] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, “On large-batch training for deep learning: Generalization gap and sharp minima,” *arXiv preprint arXiv:1609.04836*, 2016.
- [34] P. Chaudhari, A. Choromanska, S. Soatto, *et al.*, “Entropy-sgd: Biasing gradient descent into wide valleys,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2019, no. 12, p. 124018, 2019.
- [35] G. K. Dziugaite and D. M. Roy, “Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data,” *arXiv preprint arXiv:1703.11008*, 2017.
- [36] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio, “Sharp minima can generalize for deep nets,” in *International Conference on Machine Learning*, PMLR, 2017, pp. 1019–1028.
- [37] H. Wang, N. S. Keskar, C. Xiong, and R. Socher, “Identifying generalization properties in neural networks,” *arXiv preprint arXiv:1809.07402*, 2018.
- [38] D. McAllester, “Simplified pac-bayesian margin bounds,” in *Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003. Proceedings*, Springer, 2003, pp. 203–215.
- [39] B. Chugg, H. Wang, and A. Ramdas, “A unified recipe for deriving (time-uniform) pac-bayes bounds,” *arXiv preprint arXiv:2302.03421*, 2023.
- [40] M. Khan, D. Nielsen, V. Tangkaratt, W. Lin, Y. Gal, and A. Srivastava, “Fast and scalable bayesian deep learning by weight-perturbation in adam,” in *International conference on machine learning*, PMLR, 2018, pp. 2611–2620.
-

-
- [41] S. Jastrzebski, Z. Kenton, D. Arpit, *et al.*, “Three factors influencing minima in sgd,” *arXiv preprint arXiv:1711.04623*, 2017.
- [42] Z. Zhu, J. Wu, B. Yu, L. Wu, and J. Ma, “The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects,” *arXiv preprint arXiv:1803.00195*, 2018.
- [43] D. Wu, S.-T. Xia, and Y. Wang, “Adversarial weight perturbation helps robust generalization,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 2958–2969, 2020.
- [44] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, “Sharpness-aware minimization for efficiently improving generalization,” *arXiv preprint arXiv:2010.01412*, 2020.
- [45] P. Mi, L. Shen, T. Ren, *et al.*, “Make sharpness-aware minimization stronger: A sparsified perturbation approach,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 30 950–30 962, 2022.
- [46] R. A. Fisher, “On the mathematical foundations of theoretical statistics,” in *Philosophical transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, 1922, (594-604):309–368.
- [47] J. Kwon, J. Kim, H. Park, and I. K. Choi, “Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks,” in *International Conference on Machine Learning*, PMLR, 2021, pp. 5905–5914.
- [48] J. Du, H. Yan, J. Feng, *et al.*, “Efficient sharpness-aware minimization for improved training of neural networks,” *arXiv preprint arXiv:2110.03141*, 2021.
- [49] Y. Liu, S. Mai, M. Cheng, X. Chen, C.-J. Hsieh, and Y. You, “Random sharpness-aware minimization,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 543–24 556, 2022.
- [50] Y. Jiang, D. Krishnan, H. Mobahi, and S. Bengio, “Predicting the generalization gap in deep networks with margin distributions,” *arXiv preprint arXiv:1810.00113*, 2018.

-
- [51] V. N. Vapnik, *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995, ISBN: 0-387-94559-8.
- [52] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik, “Support vector regression machines,” *Advances in neural information processing systems*, vol. 9, 1996.
- [53] G. Elsayed, D. Krishnan, H. Mobahi, K. Regan, and S. Bengio, “Large margin deep networks for classification,” *Advances in neural information processing systems*, vol. 31, 2018.
- [54] K. Crammer, R. Gilad-Bachrach, A. Navot, and N. Tishby, “Margin analysis of the lvq algorithm,” *Advances in neural information processing systems*, vol. 15, 2002.
- [55] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, pp. 273–297, 1995.
- [56] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.
- [57] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky, “Spectrally-normalized margin bounds for neural networks,” *Advances in neural information processing systems*, vol. 30, 2017.
- [58] J. Sokolić, R. Giryes, G. Sapiro, and M. R. Rodrigues, “Robust large margin deep neural networks,” *IEEE Transactions on Signal Processing*, vol. 65, no. 16, pp. 4265–4280, 2017.
- [59] K. Kawaguchi, L. P. Kaelbling, and Y. Bengio, “Generalization in deep learning,” *arXiv preprint arXiv:1710.05468*, vol. 1, no. 8, 2017.
- [60] T. Poggio, K. Kawaguchi, Q. Liao, *et al.*, “Theory of deep learning iii: Explaining the non-overfitting puzzle,” *arXiv preprint arXiv:1801.00173*, 2017.
- [61] C. Mouton, M. W. Theunissen, and M. H. Davel, “Input margins can predict generalization too,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, 2024, pp. 14 379–14 387.
-

-
- [62] E. K. Blum and L. K. Li, “Approximation theory and feedforward networks,” *Neural networks*, vol. 4, no. 4, pp. 511–515, 1991.
- [63] C. K. Chui, X. Li, and H. N. Mhaskar, “Limitations of the approximation capabilities of neural networks with one hidden layer,” *Advances in Computational Mathematics*, vol. 5, no. 1, pp. 233–243, 1996.
- [64] Y. Bengio, O. Delalleau, and N. Roux, “The curse of highly variable functions for local kernel machines,” *Advances in neural information processing systems*, vol. 18, 2005.
- [65] C. K. Chui, X. Li, and H. N. Mhaskar, “Neural networks for localized approximation,” *mathematics of computation*, vol. 63, no. 208, pp. 607–623, 1994.
- [66] H. N. Mhaskar and T. Poggio, “Deep vs. shallow networks: An approximation theory perspective,” *Analysis and Applications*, vol. 14, no. 06, pp. 829–848, 2016.
- [67] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein, “On the expressive power of deep neural networks,” in *international conference on machine learning*, PMLR, 2017, pp. 2847–2854.
- [68] S.-B. Lin, “Generalization and expressivity for deep nets,” *IEEE transactions on neural networks and learning systems*, vol. 30, no. 5, pp. 1392–1406, 2019.
- [69] M. Bianchini and F. Scarselli, “On the complexity of neural network classifiers: A comparison between shallow and deep architectures,” *IEEE transactions on neural networks and learning systems*, vol. 25, no. 8, pp. 1553–1565, 2014.
- [70] J. Ba and R. Caruana, “Do deep nets really need to be deep?” *Advances in neural information processing systems*, vol. 27, 2014.
- [71] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, “The loss surfaces of multilayer networks,” in *Artificial intelligence and statistics*, PMLR, 2015, pp. 192–204.
- [72] X. Hu, L. Chu, J. Pei, W. Liu, and J. Bian, “Model complexity of deep learning: A survey,” *Knowledge and Information Systems*, vol. 63, pp. 2585–2619, 2021.
- [73] B. Neyshabur, R. Tomioka, and N. Srebro, “Norm-based capacity control in neural networks,” in *Conference on learning theory*, PMLR, 2015, pp. 1376–1401.
-

-
- [74] H. Xu and S. Mannor, “Robustness and generalization,” *Machine learning*, vol. 86, pp. 391–423, 2012.
- [75] W. Cao, Z. Yan, Z. He, and Z. He, “A comprehensive survey on geometric deep learning,” *IEEE Access*, vol. 8, pp. 35 929–35 949, 2020.
- [76] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, “Geometric deep learning: Going beyond euclidean data,” *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.
- [77] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, “Geometric deep learning: Grids, groups, graphs, geodesics, and gauges,” *arXiv preprint arXiv:2104.13478*, 2021.
- [78] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, “Geometric deep learning on graphs and manifolds using mixture model cnns,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5115–5124.
- [79] K. Atz, F. Grisoni, and G. Schneider, “Geometric deep learning on molecular representations,” *Nature Machine Intelligence*, vol. 3, no. 12, pp. 1023–1032, 2021.
- [80] P. J. Davis, *Interpolation and approximation*. Courier Corporation, 1975.
- [81] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4.
- [82] H. A. Aly and E. Dubois, “Image up-sampling using total-variation regularization with a new observation model,” *IEEE Transactions on Image Processing*, vol. 14, no. 10, pp. 1647–1659, 2005.
- [83] R. Roy, M. Pal, and T. Gulati, “Zooming digital images using interpolation techniques,” *International Journal of Application or Innovation in Engineering & Management (IJAIEEM)*, vol. 2, no. 4, pp. 34–45, 2013.
- [84] F. A. Abir, M. A. Siam, A. Sayeed, M. A. M. Hasan, and J. Shin, “Deep learning based air-writing recognition with the choice of proper interpolation technique,” *Sensors*, vol. 21, no. 24, p. 8407, 2021.
-

-
- [85] I. Chahrouh and J. Wells, “Comparing machine learning and interpolation methods for loop-level calculations,” *SciPost Physics*, vol. 12, no. 6, p. 187, 2022.
- [86] M. Belkin, S. Ma, and S. Mandal, “To understand deep learning we need to understand kernel learning,” in *International Conference on Machine Learning*, PMLR, 2018, pp. 541–549.
- [87] B. Adlam and J. Pennington, “The neural tangent kernel in high dimensions: Triple descent and a multi-scale theory of generalization,” in *International Conference on Machine Learning*, PMLR, 2020, pp. 74–84.
- [88] R. Balestriero, J. Pesenti, and Y. LeCun, “Learning in high dimension always amounts to extrapolation,” *arXiv preprint arXiv:2110.09485*, 2021.
- [89] A. Singh and T. Ogunfunmi, “An overview of variational autoencoders for source separation, finance, and bio-signal applications,” *Entropy*, vol. 24, no. 1, p. 55, 2021.
- [90] K. Kawaguchi and J. Huang, “Gradient descent finds global minima for generalizable deep neural networks of practical sizes,” in *2019 57th annual allerton conference on communication, control, and computing (Allerton)*, IEEE, 2019, pp. 92–99.
- [91] M. Davel, M. Theunissen, A. Pretorius, and E. Barnard, “Dnns as layers of cooperating classifiers,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 3725–3732.
- [92] M. D. Zeiler and R. Fergus, “Visualizing and Understanding Convolutional Networks,” in *European Conference on Computer Vision*, Springer, 2014, pp. 818–833.
- [93] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-Based Learning Applied to Document Recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [94] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.
-

-
- [95] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms,” *arXiv preprint arXiv:1708.07747*, 2017.
- [96] L. Grinsztajn, E. Oyallon, and G. Varoquaux, “Why do tree-based models still outperform deep learning on typical tabular data?” *Advances in neural information processing systems*, vol. 35, pp. 507–520, 2022.
- [97] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, “Revisiting deep learning models for tabular data,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 932–18 943, 2021.
- [98] H. J. Nussbaumer and H. J. Nussbaumer, *The fast Fourier transform*. Springer, 1982.
- [99] D. H. Wolpert, “The lack of a priori distinctions between learning algorithms,” *Neural computation*, vol. 8, no. 7, pp. 1341–1390, 1996.
- [100] E. Barnard, “Determination and the no-free-lunch paradox,” *Neural computation*, vol. 23, no. 7, pp. 1899–1909, 2011.
- [101] E. Barnard and D. Casasent, “Shift invariance and the neocognitron,” *Neural Networks*, vol. 3, no. 4, pp. 403–410, 1990.

Appendix A

Supplemental result plots

In this chapter, we provide supplementary results for the experiments conducted.

A.1 Appendix: Chapter 3

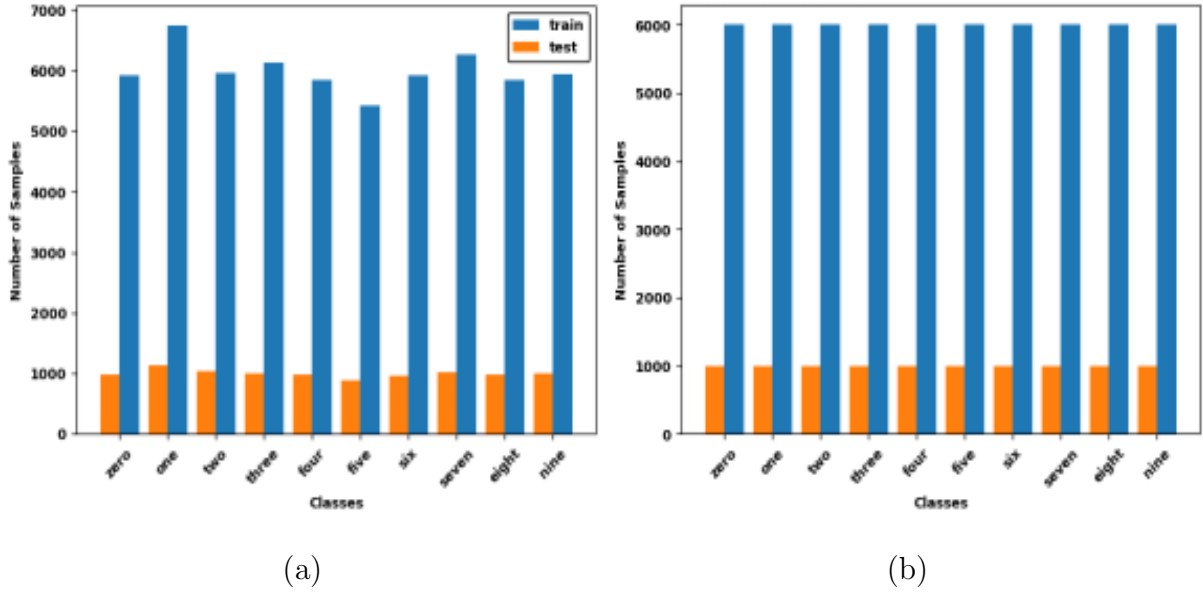


Figure A.1: a) MNIST and b) Fashion MNIST dataset class membership for training and test sets. All class members in the Fashion MNIST training and test sets are balanced while memberships in MNIST show a slight imbalance.

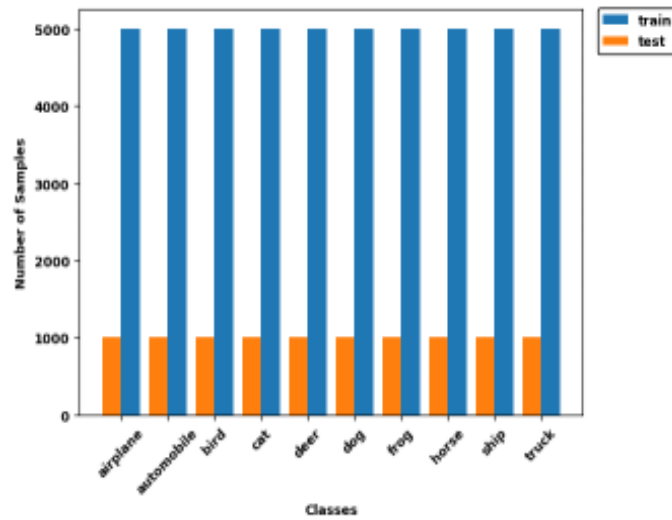


Figure A.2: CIFAR 10 dataset class membership for training and test sets. All class members in the training and test sets are balanced.

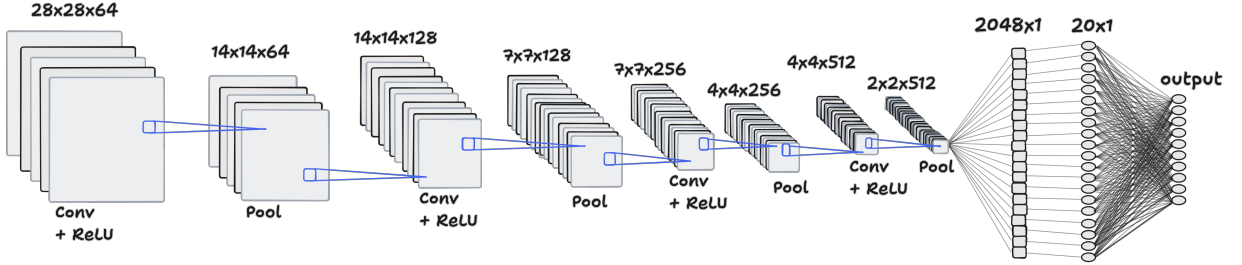


Figure A.3: Architectural design of a higher-capacity deep CNN with average pooling and a single 20-neuron fully connected layer. The network depth of 4 layers was only constrained by the input dimensions. Note that the dimensionalities shown are for FMNIST and MNIST datasets.

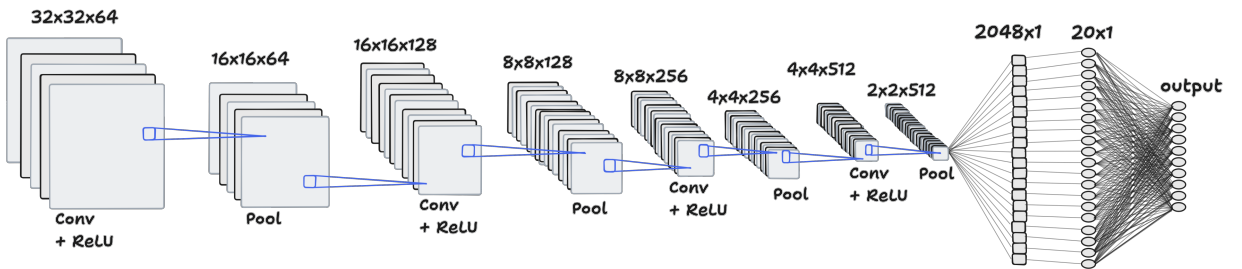


Figure A.4: architectural design of a higher-capacity deep CNN with average pooling and a single 20-neuron fully connected layer. The network depth of 4 layers was only constrained by the input dimensions. Note that the dimensionalities shown are for CIFAR-10 dataset.

A.2 Appendix: Chapter 4

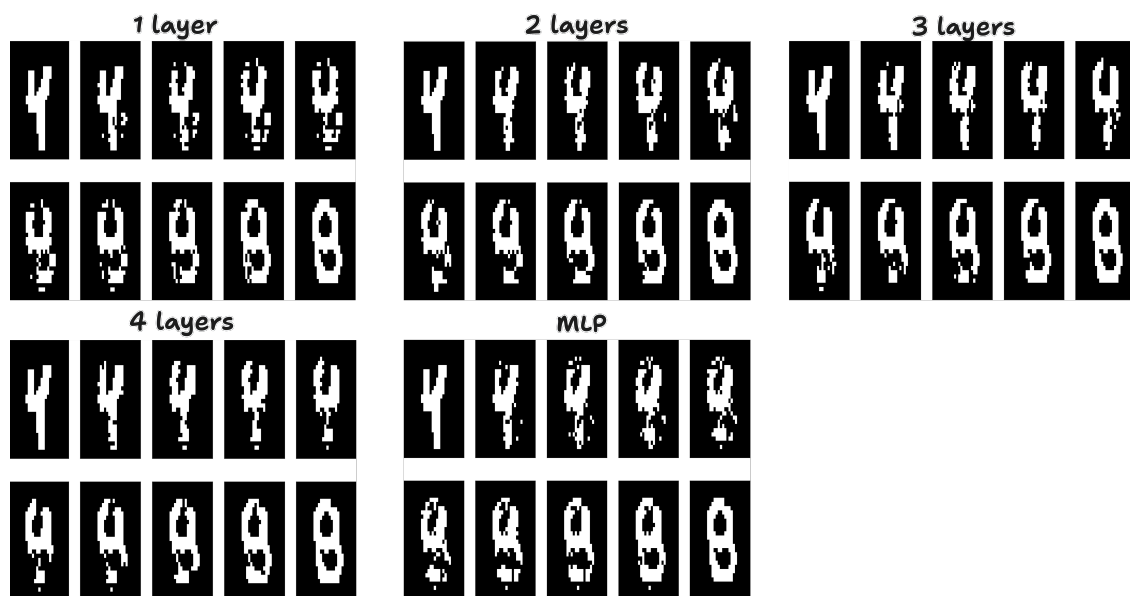


Figure A.5: MNIST classes 4 and 8 inter-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.

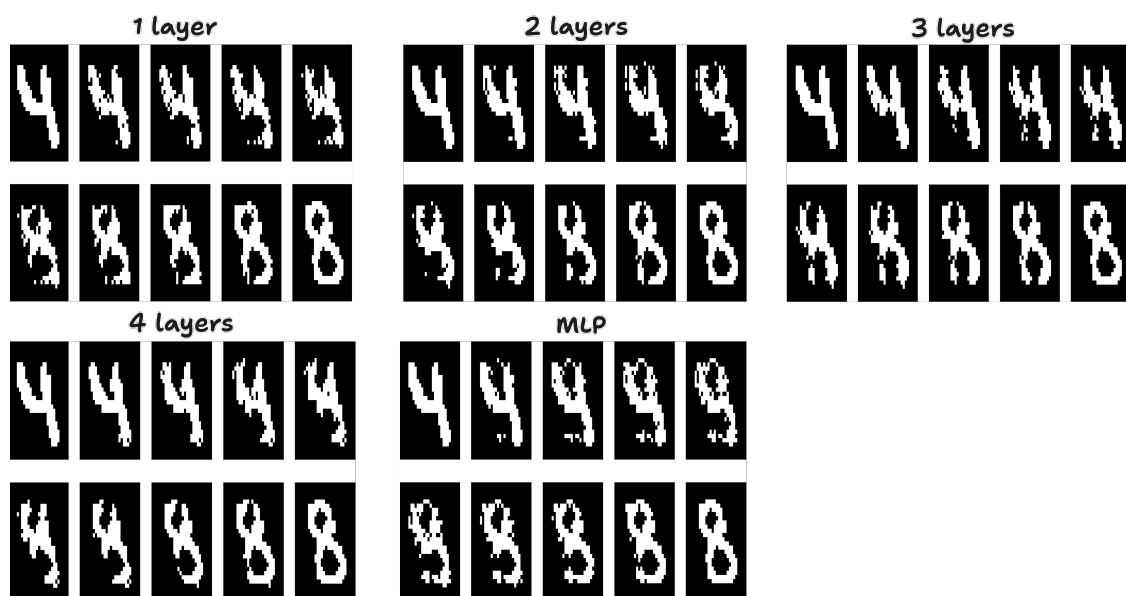


Figure A.6: MNIST classes 4 and 8 inter-class interpolation for the different CNN depths and an MLP network. Note that the layer numbers refer to the convolution layers in a given network, refer to Figure 3.1.