Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Width Drives Layers Apart

Exploring the role of critical layers in the self-regularization of neural networks

Bachelor's Thesis

Alexander Uhlmann

June 1, 2021

Advisors: Prof. Dr. Nicolai Meinshausen,
Prof. Dr. Fanny Yang, MSc Nicolò Ruggeri

Department of Mathematics, ETH Zürich

Abstract

Modern neural networks are routinely overparameterized and can easily interpolate on the training set. Yet they seem to do so in a way that generalizes well, suggesting some kind of self-regularization. In light of this, we follow up on a recent discovery that some layers of neural networks are *ambient* and can be reinitialized without incurring any significant performance drop, while others are *critical* and reduce a networks performance to guessing if reinitialized, as a possible type of self-regularization. We find evidence that width drives the separation into critical and ambient layers, which coincides with a second drop in generalization error. Furthermore, we use generative adversarial networks to create image datasets of bounded intrinsic dimension. By varying intrinsic dimension we can control training difficulty in a natural way and find that increasing training difficulty causes previously ambient layers to turn critical.

Contents

Contents **iii**

1 Introduction **1**
1.1 Related Work 2

2 Critical Layers **3**
2.1 Definitions 3
2.1.1 Setting 3
2.1.2 Reset Error and Criticality 4
2.2 Examples 5

3 Connection to Double Descent **9**
3.1 Setting 9
3.2 Double Descent in Width 11
3.2.1 Results 12
3.3 Double Descent in Training Epochs 16
3.3.1 Results 17

4 Intrinsic Dimension **21**
4.1 Generative Adversarial Networks 21
4.2 Intrinsic Dimension 22
4.2.1 Image Generation 23
4.3 Results 24
4.3.1 Dataset 24
4.3.2 Training Difficulty and Criticality 25

5 Conclusion **29**
5.1 Conjecture 29
5.2 Discussion 30
5.3 Further Directions 31
5.4 Acknowledgments 31

A Training and Model Parameters **33**
A.1 Model Architectures 33
A.2 Training Parameters 33

B GANs and Dataset Generation **35**
B.1 GANs In-depth 35
B.2 Dataset Generation 36

C Additional Plots **39**
C.1 Correlation of Test Error and Mean Reset Error 39
C.2 Layer-wise criticality for varying training epochs 41

Bibliography **43**

Chapter 1

Introduction

Deep neural networks have significantly improved the state of the art across almost all problems in machine learning. Yet in many situations large modern networks behave in a way which cannot be explained by the classical understanding of learning and achieve significantly better generalization performances than the classical bias-variance tradeoff would suggest. Zhang et al. [2017] have shown that neural networks are often heavily overparameterized and are able to easily interpolate the entire training data. This renders many traditional generalization bounds based on Rademacher complexity or VC-dimension, which both essentially measure the ability of a network to interpolate arbitrary data, vacuous [Nagarajan and Kolter, 2019].

Instead of exhibiting the U-curve predicted by the classical bias-variance tradeoff as they begin to overfit on the training data, modern overparameterized models appear to behave according to a double descent phenomenon [Nakkiran et al., 2019]: When network width is increased past the *interpolation threshold* – the point where the network is able to interpolate the entire training data – generalization error descends even further.

This suggests that as we increase width past the interpolation threshold width begins to exert some kind of regularizing force on the network which pushes it to fit the training data in a "natural" way that generalizes well. [Belkin et al., 2019, Mei and Montanari, 2019]

Although it is still unclear by what mechanism width exerts this regularizing pressure, one phenomenon which might help provide insight was discovered by Zhang et al. [2019]: In their paper the authors show that for a variety of network configurations the layers seem to differentiate into so called *critical* and *ambient* layers. When a critical layer is reset to its initial state before training it reduces network performance to random guessing, whereas an ambient layer getting reset barely impacts performance.

In this thesis, we aim to gain a better understanding of how and when these layers develop and their possible connection to double descent. We do this by analyzing critical and ambient layers under a variety of training regimes such as varying width, training epochs and datasets.

We will also make use of a novel approach for generating image datasets of bounded intrinsic dimension developed by Pope et al. [2021]. This allows us to study how critical and ambient layers adjust to varying training difficulty.

Using these methodologies, we are able to refine our understanding of layer criticality and discover some previously unknown behavior:

- Increasing width leads to a sharper separation into critical and ambient layers, but – to our surprise – does not influence *which* turn critical or ambient (see figure 3.6).
- Increasing training difficulty leads to previously ambient layers turning critical. (see figure 4.4)
- The separation into critical and ambient layer with increasing width coincides with the second descent in generalization error.(see figure 3.5)

Those results will lead us to conjecture that width drives the separation into critical and ambient layer while training difficulty determines which layers turn critical or ambient.

This thesis is structured as follows: In chapter 2, we will make formal the notions needed to talk about critical and ambient layers and then observe their behavior in a number of examples. In chapter 3, we will describe the phenomenon of double descent and see how critical layers relate to it by varying both network width and then the number of training epochs. In chapter 4, we will generate images with bounded intrinsic dimension and explore how this affects training difficulty and critical layers. And finally in chapter 5, we will tie together the results from the previous chapter into a conjecture and discuss possible explanations and ramifications.

1.1 Related Work

Chatterji et al. [2020] refine the notion of critical and ambient layers: By not only measuring the accuracy drop if a layer is reset but observing network performance if a layer is linearly moved from its final state back to initialization they develop a measure called *module criticality*. The authors then show that this measure is able to predict generalization performance more accurately than previously proposed measures.

The fact that many neural networks are overparameterized has also been exploited by Han et al. [2016] to compress models after training – achieving significant speedup without incurring a drop in performance.

Chapter 2

Critical Layers

In this chapter we will first define some key concepts to make the notions of critical and ambient layers clear. We will then give examples of criticality in a real-world setting and see that simple explanations are insufficient to describe this phenomenon.

2.1 Definitions

2.1.1 Setting

We will begin by outlining the setting and notation used for the training of models in this thesis. As we will only be training models for classification the procedure will always be:

- Our data consists of train set $\mathcal{R}$ and a test set $\mathcal{S}$ drawn from an underlying distribution $\mathcal{D}_{data}$:

$$\mathcal{R} = \{(\mathbf{x}^{(i)}, y^{(i)}) \sim \mathcal{D}_{data} | \; i \text{ in } 1, \dots, n\} \subset \mathbb{R}^k \times \mathbb{R}$$
$$\mathcal{T} = \{(\mathbf{x}^{(i)}, y^{(i)}) \sim \mathcal{D}_{data} | \; i \text{ in } 1, \dots, m\} \subset \mathbb{R}^k \times \mathbb{R}$$

  Note that n and m denote the number of train and test samples respectively while k is the dimension of the input the network will receive. For our purposes we will represent images as flattened to a vector.

- A network $f_{\theta^{(0)}} : \mathbb{R}^k \to \mathbb{R}$ is then created and initialized with $\theta^{(0)} = (\theta_0^{(0)}, \dots, \theta_D^{(0)})$ containing the parameters of each layer at initialization. The parameters of each layer are initialized by sampling from some initial distribution $\theta_d^{(0)} \sim \mathcal{D}_{init,\, d}$ which depends on the type of layer as well as the shape of inputs and outputs.

- We then train the network for a total of T epochs with $\theta^{(t)}$ representing the network parameters after the t-th Epoch. The final network is then represented as $f_{\theta^{(T)}}$.

We can evaluate the performance of a model by calculating the classification error on a given set of samples:

Definition 2.1 (Classification Error) *Let f_θ be a model and $\mathcal{M}$ a set of samples. The* error *of f_θ on $\mathcal{M}$ is then defined to be:*

$$\text{Error}_{\mathcal{M}}(f_\theta) := \frac{|\{(\mathbf{x}, y) \in \mathcal{M} | f_\theta(\mathbf{x}) \neq y\}|}{|\mathcal{M}|} \in [0,1]$$

In particular we will call $\text{Error}_{\mathcal{R}}(f_\theta)$ the *train error* of f_θ and $\text{Error}_{\mathcal{S}}(f_\theta)$ the *test error* or *generalization error* of f_θ.

2.1.2 Reset Error and Criticality

Each layer applies some nonlinear transformation to the output of the previous layer. As a result neural networks are able to iteratively build up more and more abstract representation of the input data. The recursive nature of this process makes it difficult to determine which calculations each layer performs, but one way to assess the importance of a particular layer to the networks functioning is to look at the performance drop one incurs when resetting a layer to its original state:

Definition 2.2 (Reset Error) *Let $f_{\theta^{(T)}}$ be a network trained for T epochs. We now reset the parameters of layer d to their state at initialization: $\theta_d^{(T)} \leftarrow \theta_d^{(0)}$. The resulting network $f_{\theta'}$ then has parameters*

$$\theta' = (\theta_1^{(T)}, \ldots, \theta_{d-1}^{(T)}, \theta_d^{(0)}, \theta_{d+1}^{(T)}, \ldots, \theta_D^{(T)})$$

The reset error is now obtained by evaluating the test error of the modified network:

$$\text{ResetError}(f_{\theta^{(T)}}, d) := \text{Error}_{\mathcal{S}}(f_{\theta'})$$

As the test error itself is bounded inside $[0,1]$ so is the reset error of any layer. In reality we can say even more:

Remark 2.3 (Heuristic Reset Error Bounds) *Since it is unlikely that resetting a layer will* improve *test performance we get that*

$$\text{ResetError}(f_{\theta^{(T)}}, d) \lesssim \text{Error}_{\text{S}}(f_{\theta^{(T)}}) \quad \textit{for } d = 1, \ldots, D$$

Conversely it is also unlikely – especially if the dataset classes are uniformly distributed – that resetting a layer will make the test performance drop below just randomly guessing:

$$\text{ResetError}(f_{\theta^{(T)}}, d) \gtrsim 1 - \frac{1}{c} \quad \textit{for } d = 1, \ldots, D$$

where c denotes the number of classes in the dataset.

We will now define critical and ambient layers to be layers that have a reset error close one of these extremes:

Definition 2.4 (Critical and Ambient Layers) *Let $\varepsilon > 0$ be some small parameter. The d-th layer of a network is called* ambient *if resetting it does not significantly decrease performance:*

$$\text{ResetError}(f_{\theta^{(T)}}, d) > \text{Error}_S(f_{\theta^{(T)}}) - \varepsilon$$

Conversely a layer is called critical *if resetting it reduces the networks performance to random guessing:*

$$\text{ResetError}(f_{\theta^{(T)}}, d) < (1 - \frac{1}{c}) + \varepsilon$$

Note that this still doesn't rigorously define critical and ambient layers since ε has to be chosen qualitatively. In practice choosing $\varepsilon = \frac{1}{10}$ is appropriate unless the test error of the model is very high which reduces the range of the reset error too far.

In line with these definitions we will call layers with a higher reset error more *critical*.

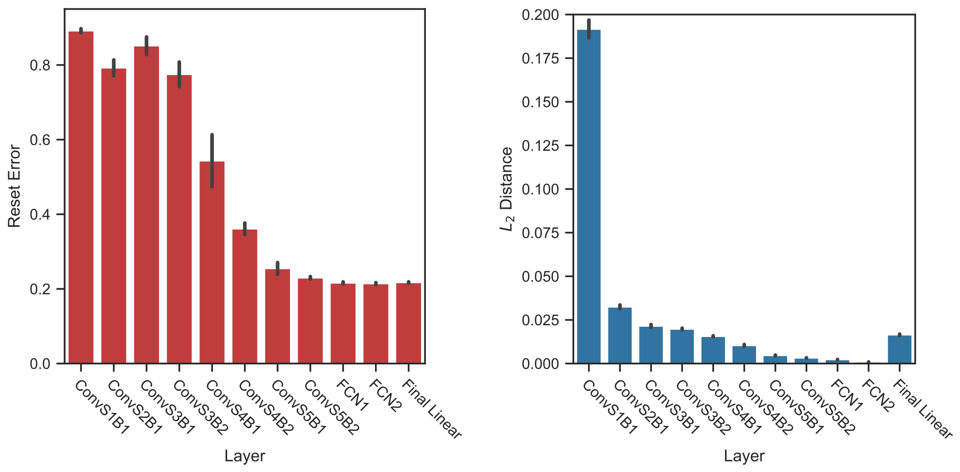
To qualitatively analyze the distribution of critical and ambient layers we will often look at the mean and variance of the reset error across layers:

- The mean reset error decreases as layers become more ambient overall. We will see that it correlates with test error.
- The variance of the reset error is often more interesting since a higher variance signifies a sharper separation into critical and ambient layers. It is maximized if half of the layers are fully critical while the other half are fully ambient.

2.2 Examples

Equipped with these definitions we can now examine how they hold up in a real-world setting. A priori one would maybe guess that all layers are fully critical since the representations a network develops during training are so highly dependent on each other that resetting just one layer would completely ruin the network performance. Another reasonable expectation would be that resetting a layer would just decrease the network performance by some fixed amount. As we will see this indeed seems to be the case in the early stages of training.

However, as we can see in figure 2.1 a VGG11 network [Simonyan and Zisserman, 2015] fully trained on CIFAR10 [Krizhevsky et al., 2009] splits almost



(a) Layer-wise reset error

(b) Layer-wise euclidean distance from initialization

Figure 2.1: **Reset error and distance from initialization for VGG11 trained on CIFAR10.**

completely into critical and ambient layers: The first four convolutional layers are almost fully critical while resetting the last four layers barely impacts performance.

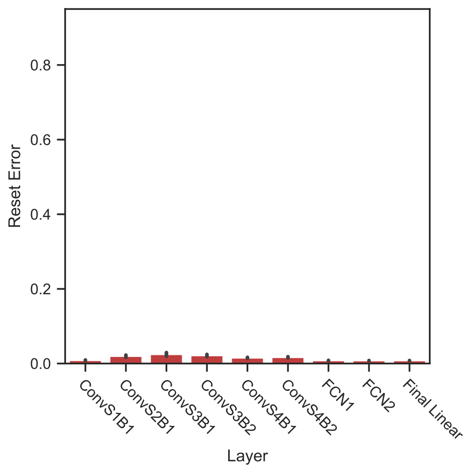
Another reasonable explanation for the existence of critical and ambient layers might be that the parameters of critical layers might simply have moved further away from their value at initialization. To inspect this explanation we will look at the normalized euclidean distance of a fully trained layer from its state at initialization:

$$\frac{1}{\sqrt{p}} \|\theta_d^{(T)} - \theta_d^{(0)}\|_2 \quad \text{for } \theta_d^{(T)}, \theta_d^{(0)} \in \mathbb{R}^p$$

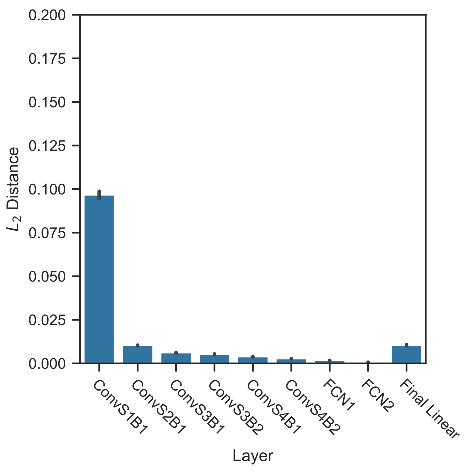
where p is the number of parameters of the layer.

But as we can see in figure 2.1 euclidean distance from initialization doesn't seem to correlate with the reset error. Especially the first and last layer have moved disproportionately large distances from initialization compared to their respective reset error.

If we now compare these results to those of a network trained on the significantly easier MNIST dataset [LeCun and Cortes, 2010], the lack of correlation between euclidean distance from initialization and reset error becomes even clearer. In figure 2.2 we can see that in contrast to the network trained on CIFAR10, all layers are almost completely ambient, reflecting the significantly



(a) Layer-wise reset error



(b) Layer-wise euclidean distance from initialization

Figure 2.2: **Reset error and distance from initialization for a reduced VGG11 trained on MNIST.** See appendix A for details regarding the reduced VGG11 architecture.

easier training task. However, even though all layers are ambient, some have still moved a significant amount from initialization.

Chapter 3

Connection to Double Descent

We will begin this chapter by taking a more in-depth look at double descent and introduce the notions required to describe it more precisely. We will then turn our attention to the connection between double descent and layer criticality by first observing double descent in varying width and then in varying training epochs.

3.1 Setting

As previously mentioned, neural networks seem to defy the classical understanding of the bias-variance tradeoff. Large models easily capable of interpolating the entire training data do so in a way which generalizes well. This by itself is a non trivial observation. As the function class represented by highly overparameterized neural networks is intuitively large, other interpolating but non generalizing, solutions exist. Convergence to good solutions is therefore not to be given for granted. In settings with noisy data (which pushes models to overfit), a *double descent* of test error can be observed [Nakkiran et al., 2019]: As model capacity increases we first observe the classical bias-variance tradeoff with test error reaching a peak at the *interpolation threshold* – the point where the network is able to interpolate the entire training data. But if we keep increasing the model capacity, the test error begins to decrease again – hence double descent (see figure 3.1 for a schematic representation).

While it is unclear what causes the second descent, a common intuition is that if model capacity is increased past the point where the model is able to just *barely* fit the training data, then the model is able to fit the training data in a more "natural" way [Belkin et al., 2019]. Of course the notion of a "natural" way to fit data is fuzzy and the question remains what pushes networks to fit in such a "natural" way as capacity increases.

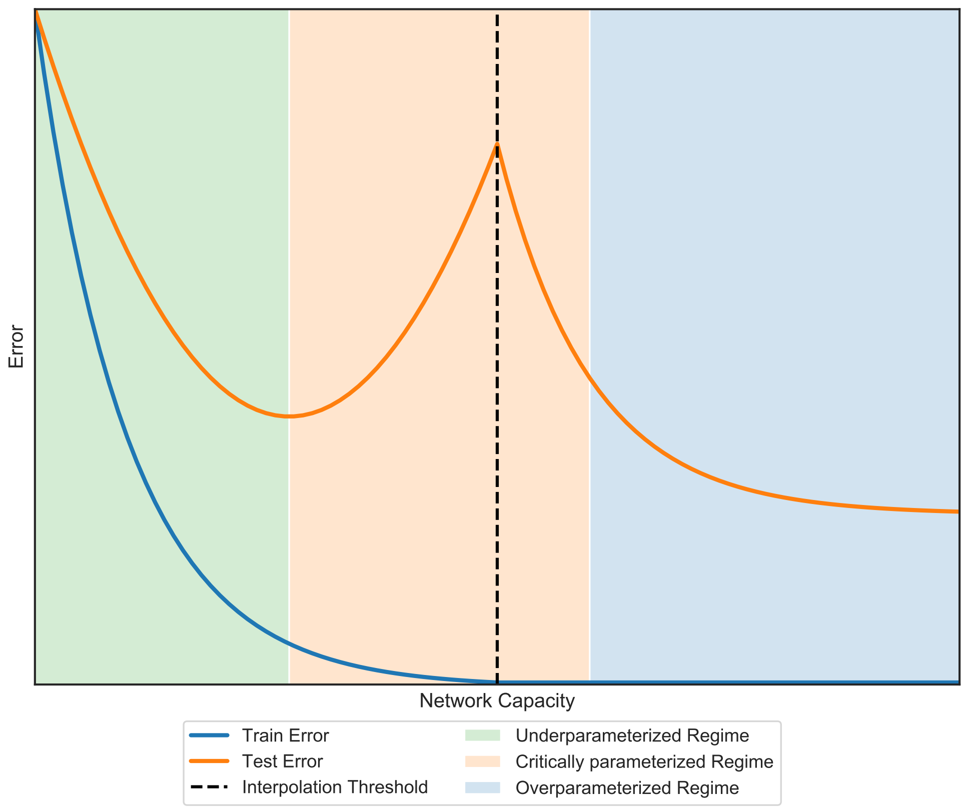


Figure 3.1: **Schematic representation of the *double descent* train and test error curves.** It incorporates the classical bias-variance U-shaped test error curve in the first half and the modern "larger models are better"-test error curve in the second half. Note that the test error reaches its second peak at the interpolation threshold

As previously mentioned we will now look at the notions required to describe this phenomenon precisely. First of all let us make formal notion of *network capacity* following the definitions developed by Nakkiran et al. [2019]:

We define a *training procedure* $\mathcal{T}$ to be any procedure which takes a training dataset $\mathcal{R} \sim \mathcal{D}^n$ and returns a trained classifier $\mathcal{T}(\mathcal{R})$. For any given training procedure $\mathcal{T}$ and data distribution $\mathcal{D}$ and we define the *Effective Model Capacity (EMC)* to be the maximal number of training samples on which $\mathcal{T}$ is able to achieve near zero error:

Definition 3.1 *The* Effective Model Capacity (EMC) *of a training procedure $\mathcal{T}$ with respect to distribution $\mathcal{D}$ and parameter $\varepsilon > 0$ is defined as:*

$$\mathrm{EMC}_{\mathcal{D},\varepsilon}(\mathcal{S}) := \max\{n | \mathbb{E}_{\mathcal{R}\sim\mathcal{D}^n}[\mathrm{Error}_{\mathcal{R}}(\mathcal{T}(\mathcal{S}))] < \varepsilon\}$$

Note that while our notion of capacity is related to classical measures of capacity such as Rademacher complexity and VC-dimension [Bartlett and Mendelson, 2002, Vapnik, 2013], it differs in some significant ways: Instead of being determined model architecture alone, EMC also depends on the sample distribution and the training procedure – making it impossible to determine *a priori*. These dependencies are however necessary to accurately describe the dynamics of double descent [Nakkiran et al., 2019].

We can now more precisely specify double descent, again following Nakkiran et al. [2019]:

Conjecture 1 *For any natural data distribution $\mathcal{D}$, neural net based training procedure $\mathcal{T}$, training set $\mathcal{R} \sim \mathcal{D}^n$ and small $\varepsilon > 0$, the relationship between test error and $\mathrm{EMC}_{\mathcal{D},\varepsilon}(\mathcal{S})$ can be split into three regimes:*

Underparameterized regime: *If $\mathrm{EMC}_{\mathcal{D},\varepsilon}(\mathcal{S})$ is sufficiently smaller than the number of training samples n, any perturbation of $\mathcal{T}$ that increases its effective capacity will* decrease *the test error.*

Overparameterized regime: *If $\mathrm{EMC}_{\mathcal{D},\varepsilon}(\mathcal{S})$ is sufficiently larger than the number of training samples n, any perturbation of $\mathcal{T}$ that increases its effective capacity will* decrease *the test error.*

Critically parameterized regime: *If $\mathrm{EMC}_{\mathcal{D},\varepsilon}(\mathcal{S}) \approx n$, a perturbation of $\mathcal{T}$ that increases its effective capacity might* increase or decrease *test error.*

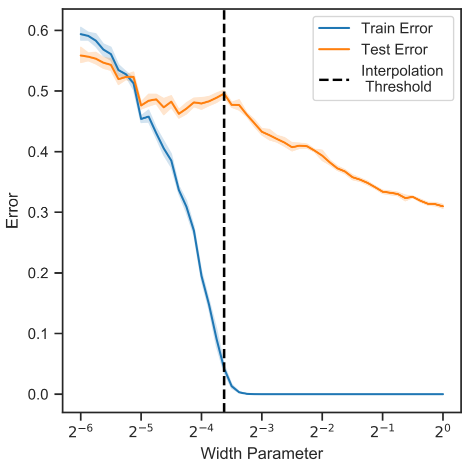
These regimes are illustrated in figure 3.1 and we can see that the underparameterized regime corresponds to the first descent in test error, the critically-parameterized regime contains the region where the model begins to overfit and the test error reaches its second peak and the over-parameterized regime contains the second descent in test error.

Two ways of controlling EMC are varying the network width and varying the number of training epochs. In the next sections we will see that in both cases the separation of layers into critical and ambient seems to sharpen with increasing model capacity.

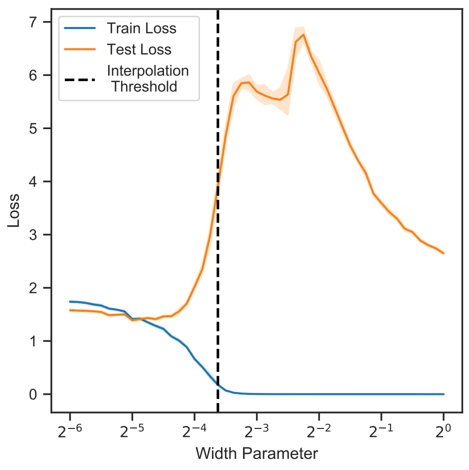
3.2 Double Descent in Width

To study the effects of width on layer criticality we will vary the width of our models by introducing a *width parameter* $s \in \mathbb{R}_{>0}$. The scaled model is then obtained by scaling the width of every layer in the base model (in our case VGG11 – see appendix A for more details) by the width parameter and rounding to the closest integer width.

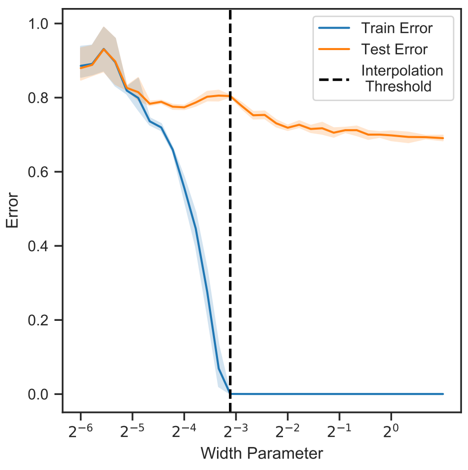
To heighten the effects of double descent we introduce 15% label noise since double descent is a product of temporary overfitting and label noise leads a



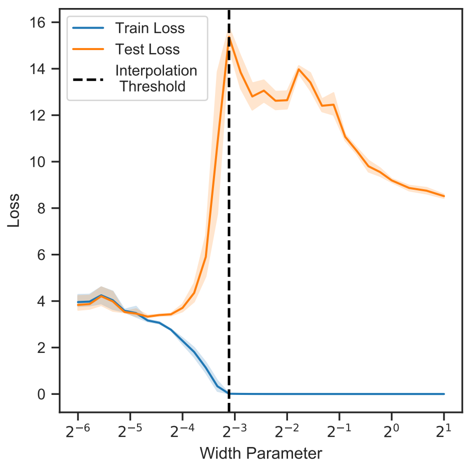
(a) Train and test error on CIFAR10



(b) Train and test loss on CIFAR10



(c) Train and test error on CIFAR100



(d) Train and test loss on CIFAR100

Figure 3.2: **Training metrics as a function of model width.** Models are of VGG11 architecture and are trained on CIFAR10 and CIFAR100 with 15% label noise added. The interpolation thresholds were determined qualitatively.

model to overfit more extremely. This is in line with the results of Nakkiran et al. [2019], who observed that more label noise leads to more extreme double descent peaks. To compare the effects of scaling network width on datasets with varying difficulty we will compare models trained on CIFAR10 and CIFAR100.

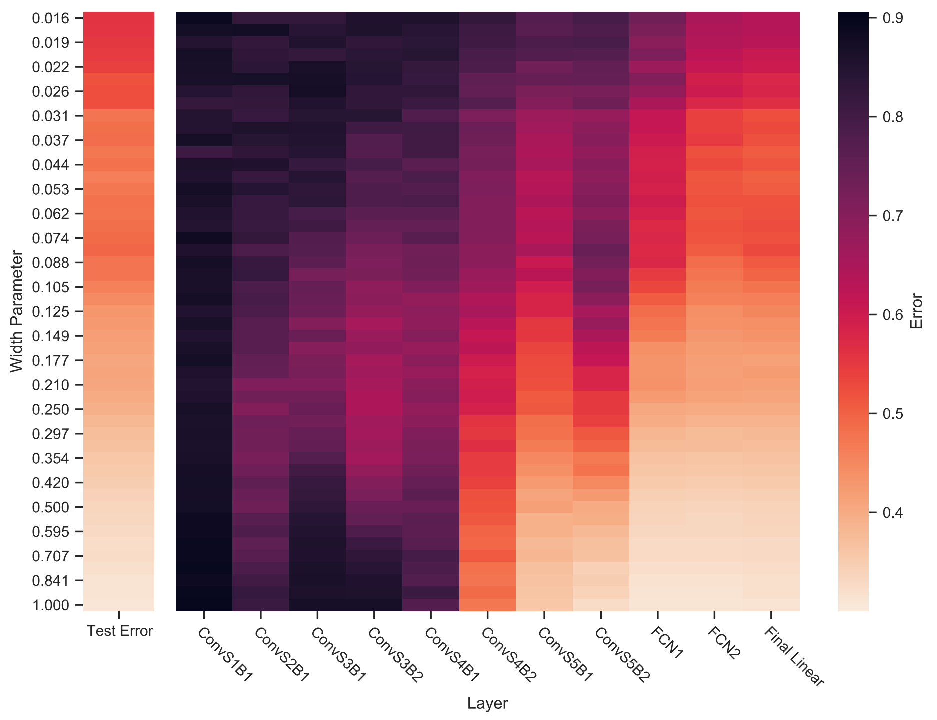


Figure 3.3: **Layer-wise reset error for varying width on CIFAR10.** The leftmost column contains the test errors and every other column corresponds to the reset errors of a given layer for varying model width. Each row corresponds to a specific width parameter. Critical layers turn black and ambient layers turn white.

3.2.1 Results

As we can see in figure 3.2, a double descent in width does indeed occur with models trained on CIFAR10. The effect is even more extreme if we look at test loss instead of test error. We can observe the same for CIFAR100 (see figure 3.2), although it takes a slightly higher width for the model to reach the interpolation threshold. While the training sets of CIFAR10 and CIFAR100 both contain 60′000 of samples, it makes intuitively sense that interpolating samples belonging to 100 classes would require more capacity than doing the same for 10 classes.

If we now turn our attention to the layer criticalities in figures 3.3 and 3.4, we can see that as model width increases we can observe an interesting phenomenon: We can see a separation in the layers of the models into *critical* layers which turn fully critical with increasing width and *ambient* layers which turn fully ambient with increasing width. As width increases we see the separation between these critical and ambient layers get more sharp.

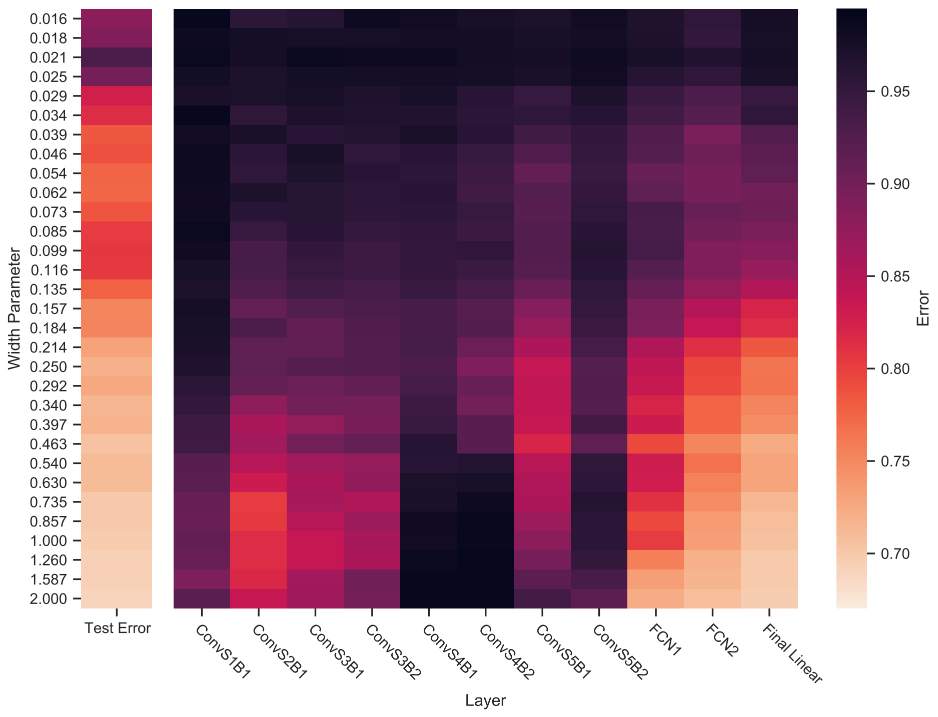


Figure 3.4: **Layer-wise reset error for varying width on CIFAR100.** Same layout as figure 3.3

We can also observe that although almost all layers develop into either a fully critical or fully ambient layer, for some it takes significantly higher model width than for others. Furthermore, we can observe that many critical layers first go through a phase of decreasing reset error before it begins to *increase* again and the layer turns fully critical.

The separation into critical and ambient layer is even clearer if we look at the increase in reset error variance across layers in figure 3.5. Note that the separation only seems to occur *after* the interpolation threshold has been passed. Moreover we can see that, while the variance behaves in an interesting way, the mean reset error seems to roughly correlate with test error. This is not surprising since, as discussed in chapter 2, the reset error is often bounded from by the test error. So as the test error decreases, the range of possible reset errors expands further down and the mean reset error decreases as well. This correlation is further explored in appendix C.

Figures 3.3 and 3.4 allow us to qualitatively separate the layers of our models based on whether they turn more critical or more ambient with increasing width. We see that for the model trained on CIFAR10 the first five layers turn critical and the following six turn ambient, for CIFAR100 all but the last

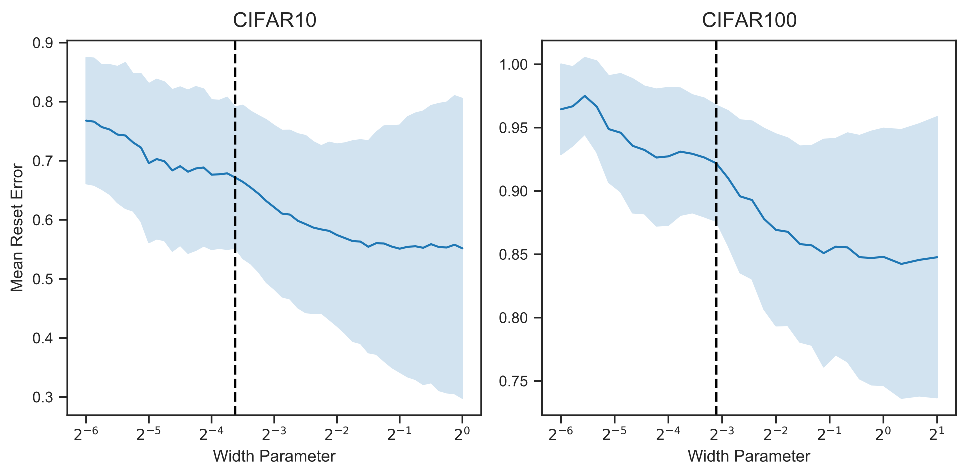


Figure 3.5: **Mean and variance of the reset error across layers as a function of model width.**

three layers turn critical.

In figure 3.6 the separation into fully critical and fully ambient layers is highlighted by coloring layers which turn critical and layers which turn ambient differently. We can see that for CIFAR10, at a width parameter of 1 the layers are almost completely split into fully critical and ambient layers. For CIFAR100 this separation takes significantly longer: Even at a width parameter of 2 the separation is still not complete with some layers just starting to turn fully critical.

The behavior of layer criticality, as observed in figures 3.3, 3.4, 3.5 and 3.6, can thus be summarized as follows:

- For widths leaving model capacity below the interpolation threshold, the reset error of all layers seems to roughly correlate with the test error – while resetting some layers results in a bigger absolute drop in model performance, the shape remains similar.
- But as the width increases past the interpolation threshold we see a separation of layer criticality, with layers either turning fully critical or fully ambient.
- In particular: For some layers the reset error which initially decreased as it moved in tandem with the improving test error, begins to increase again as the layer turns fully critical.

Especially the last observation is really surprising: We initially expected that increasing width would lead to previously critical layers turning ambient one by one as the network "turns off" layers which are no longer needed.

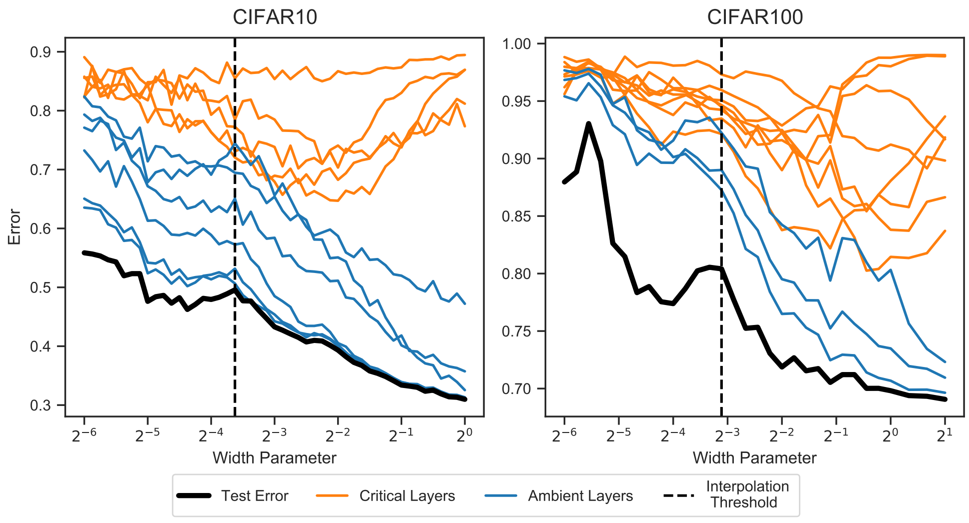


Figure 3.6: **Layer-wise criticality and test error as a function of model width.** The test error is drawn in black and every other line represents the reset error of a specific layer. Layers are colored qualitatively based on whether they are *critical* and turn more critical as width increases or are *ambient* and turn more ambient as width increases.

However, as we can see in figure 3.6 this is not the case and it seems that width drives the separation into fully critical and fully ambient layers, but has no influence over *which* layer become critical or ambient. Instead, if we look at the key differences between the models trained CIFAR10 and CIFAR100, which layers turn critical seems to be determined by training difficulty:

- While we can see a differentiation into critical and ambient layers in both cases, it takes significantly longer for this to happen in the case of the more difficult CIFAR100, even after we scale model width up to a factor of 2.

- Training difficulty also seems to not only impact the width required for layers to fully separate into critical and ambient, but also *which* layers turn critical or ambient in the first place: In the case of CIFAR10 6 layers turn ambient, while for the harder CIFAR100 only 3 layers turn ambient.

In chapter 4 we will examine the effects of training difficulty more closely which will lead us to hypothesize that while width drives the separation into critical and ambient layers, it is training difficulty which determines which layers turn critical or ambient. We will then unify these findings in chapter 5.

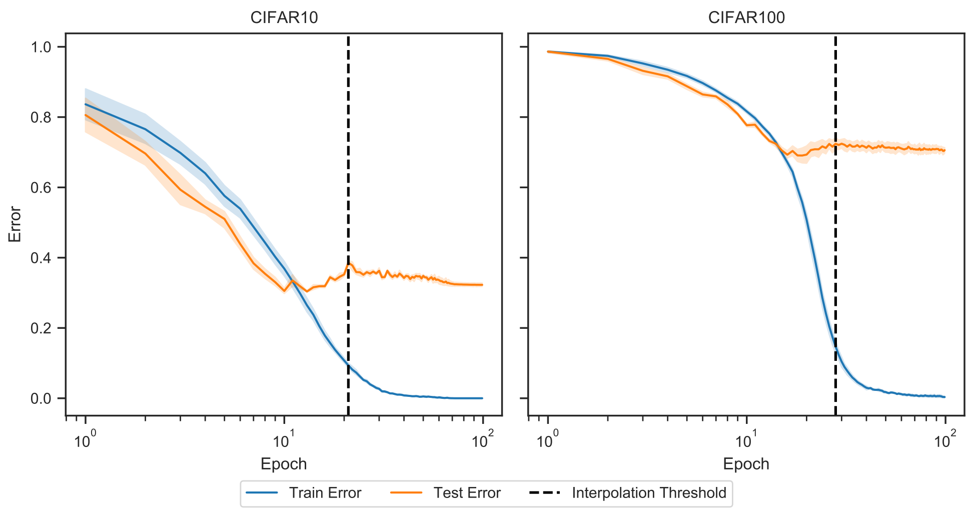


Figure 3.7: **Train and test error of models trained on CIFAR10 and CIFAR100 respectively as a function of training epochs.** Models are again VGG11 and 15% label noise was added.

3.3 Double Descent in Training Epochs

We now turn our attention to the other main hyperparameter related to double descent identified by Nakkiran et al. [2019]: The number of training epochs.

We will measure the effect the number of training epochs has on layer criticality by evaluating the reset error after every epoch. To highlight double descent we again introduced 15% label noise. We then trained models on CIFAR10 and CIFAR100 for 100 epochs. Furthermore we also trained a model on the much easier MNIST dataset, this will allow us to investigate layer criticality for the case of all layers turning ambient.

3.3.1 Results

CIFAR10 and CIFAR100

In figure 3.7 we can see that for both datasets a double descent of test error in training epochs occurs, although for CIFAR100 the effect is rather weak.

If we now look at layer criticality by measuring mean and variance of reset error we can see that a similar pattern as in the case of varying width emerges: In 3.8 we can see that the average reset error again correlates with the test error and that the variance in reset error increases with the number of training epochs.

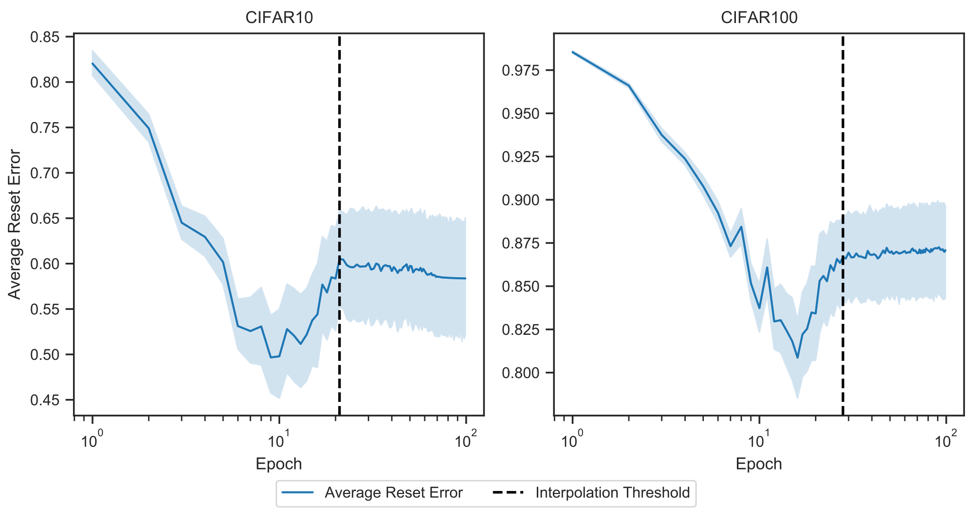


Figure 3.8: **Mean and variance of the reset error across layers as a function of training epochs.**

We can also see that the behaviour for CIFAR10 and CIFAR100 is very similar. The only notable difference is that models trained on CIFAR100 reach the interpolation threshold about 10 epochs later. This is in line with the observation in the previous section where a model trained on CIFAR100 required more width to reach the interpolation threshold.

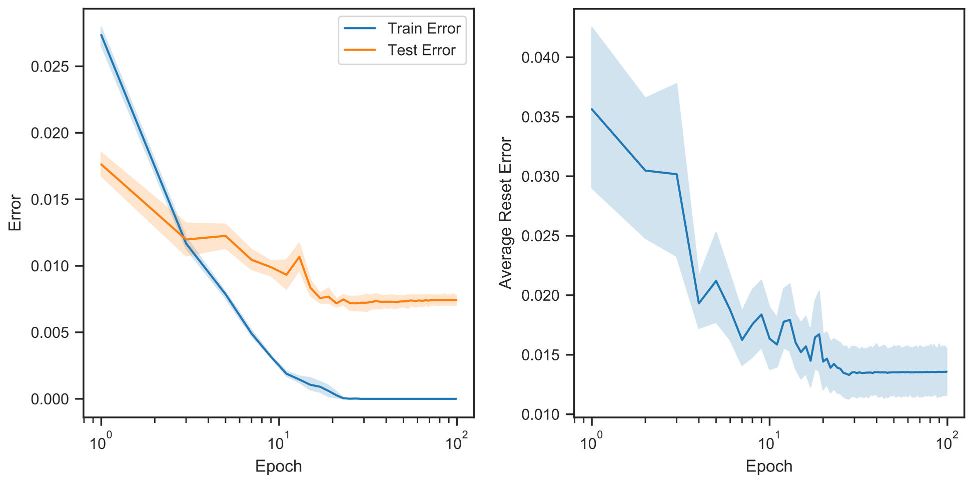
There are however also some key differences in the case of varying training epochs compared to the case of varying width:

- The separation into critical and ambient layers seems to occur *before* model capacity reaches the interpolation threshold.
- Past the point where the models are able to reach the interpolation threshold, very little change in layer criticality seems to occur.

We will examine these results more closely in appendix C.

The fact that layer criticality changes very little if we continue training past the interpolation threshold is not surprising considering the following: A model reaching the interpolation threshold is able to achieve a very low training error. This means that in most cases the training loss is also going to be very low. And since training loss is what drives change in layers, we would expect layers in general – and layer criticality in particular – to change very little past the interpolation threshold. This is mirrored in the results of Nakkiran et al. [2019], where double descent in width is stronger than double descent in learning epochs.

If we accept that very little change in criticality will happen past the interpola-



(a) Train and test error

(b) Mean and variance of the reset error across layers

Figure 3.9: **Training and layer criticality metrics as a function of training epochs for models trained on MNIST.** Models are reduced VGG11 (see appendix A), no training noise was added.

tion threshold, the first observation becomes less surprising as well: We know that by the end of training, a model of given width will have developed a certain separation of critical and ambient layers, and since criticality changes very little past the interpolation threshold, the separation has to occur before that.

Based on those findings we can hypothesize that while increasing the number of training epochs past the interpolation threshold doesn't cause layers to further separate into critical and ambient, training the model for too few epochs can prevent the layers from separating. Training models for an adequate number of epochs is thus a necessary but not sufficient condition for the separation of layers to occur.

MNIST

Finally, we will compare the results of models trained on CIFAR10 and CIFAR100 to those of models trained on the significantly easier MNIST dataset. Note that since models trained on MNIST are able to interpolate the training set almost immediately we cannot determine a sensible interpolation threshold.

As we can see in figure 3.9, average reset error again moves in tandem with reset error. There is however a significant difference to the earlier cases of models trained on CIFAR10 and CIFAR100: As training time increases, we see the variance in reset error go *down*. This is because for models trained on MNIST, as we saw in figure 2.2, all layers are ambient and the decreasing variance is therefore the result of all layers turning fully ambient.

Chapter 4

Intrinsic Dimension

A common intuition in computer vision is that while natural image data may be of high ambient dimension, it actually lies on an embedded manifold of significantly lower dimension, also known as *intrinsic dimension*. There is mounting evidence which suggests that not ambient, but intrinsic dimension of the data is a key determinant of sample complexity and learning difficulty [Pope et al., 2021, Narayanan and Mitter, 2010].

To investigate how layer criticality behaves under varying training complexity, we will use a generative adversarial network (GAN) to generate datasets with a bounded intrinsic dimension, following a novel method developed by Pope et al. [2021]. This will allow us to compare the development of critical layers in networks trained on datasets of varying intrinsic dimension. We will see that dataset of higher intrinsic dimension are indeed more challenging, which leads networks to develop more critical layers.

4.1 Generative Adversarial Networks

Generative adversarial networks are a class of machine learning frameworks introduced by Goodfellow et al. [2014]. In recent years they have been remarkably successful at generating images mimicking those in a given dataset. In this section we will give an overview of their structure and introduce the BigGAN models developed by Brock et al. [2019], as we will be using these models to generate images of bounded intrinsic dimension in the next section.

GANs consist of two networks: A generator G tasked with generating images mimicking those in the original dataset and a discriminator D tasked with distinguishing between generated and real images. These two networks are then trained simultaneously and as the discriminator gets better at identifying generated images, the generator begins to produce images which closely

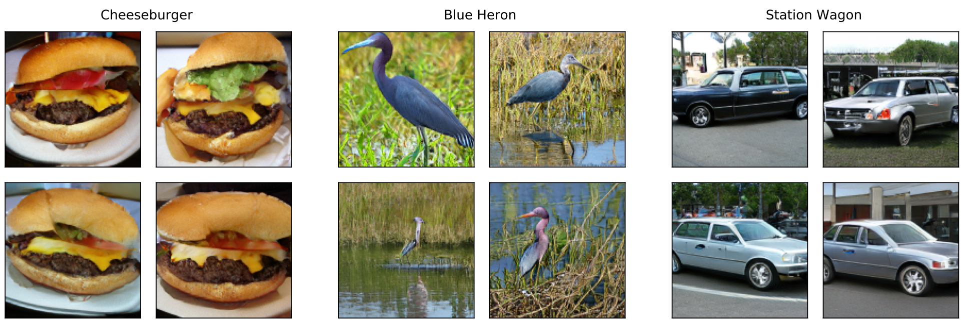


Figure 4.1: **Sample images of three ImageNet categories generated by a BigGAN128 model.**

resemble those in the original dataset. The generator synthesizes images by mapping some noise vector $\mathbf{z} \in \mathbb{R}^m \sim \mathcal{D}_{noise}$ to an image $G(\mathbf{z}) \in \mathbb{R}^k$ mimicking the real ones drawn from the data distribution $\mathcal{D}_{data}$. A more in-depth description of this process can be found in appendix B.

For our purposes we will use the BigGAN Models developed by Brock et al. [2019]. These GANs have been trained to generate samples mimicking the ImageNet dataset [Deng et al., 2009]. Since the training process of GANs is notoriously brittle and prone to collapse, a number of regularisation techniques were introduced. Additionally, scaling up both batch size and model size led to more stable and effective training. As a result the BigGAN models are able to generate high quality images of resolutions up to 512x512 pixels (see figure 4.1 for examples).

Another property of the BigGAN architecture which is very useful, is that BigGANs are able to generate class conditional samples. To that end the generator G takes an additional vector $\mathbf{c} \in [0,1]^c$, where c is the number of classes, as input. Each component of this vector $\mathbf{c}$ encodes the probability of the desired image belonging to the respective class.

4.2 Intrinsic Dimension

As previously mentioned, the dimensionality of a dataset plays an important role in determining the difficulty of the training task. To give an example, Narayanan and Mitter [2010] show that learning a manifold requires a number of samples which increases exponentially in manifold dimension. An intuition for this can be gained by considering the hypercube in d dimensions, for which sampling only the vertices requires 2^d measurements.

Fortunately, in practice many datasets with high ambient dimension appear to lie on a much smaller dimensional embedded manifold. We will now make this notion formal:

Definition 4.1 (Intrinsic Dimension) *A set of images $\mathcal{C} \subset \mathbb{R}^k$ is said to have* intrinsic dimension $\bar{d}$ *if there exist a manifold $\mathcal{M}$ with dimension $\bar{d} := \dim \mathcal{M}$ and an embedding $\iota : \mathcal{M} \to \mathbb{R}^k$ such that for every image $\mathbf{x} \in \mathcal{C}$ there exists a point $p \in \mathcal{M}$ with $\iota(p) = \mathbf{x}$.*

As previously mentioned, the dimension which affects training difficulty is the intrinsic dimension and not the dimension of the dataset space [Narayanan and Mitter, 2010]. In particular for the case of image data, Pope et al. [2021] show that intrinsic dimension and not image resolution determine sample complexity.

4.2.1 Image Generation

We will now introduce the method developed by Pope et al. [2021] to generate image sets with a bounded intrinsic dimension:

The main idea is that by setting all but $\bar{d}$ entries in the noise vector $\mathbf{z} \sim \mathcal{D}_{noise}$ to zero, we can guarantee that, for a given class $\mathbf{c} \in \mathbb{R}^c$, the image created by the generator $G(\mathbf{z}, \mathbf{c})$ lies on a manifold of at most dimension $\bar{d}$ since G is a Lipschitz continuous function which cannot increase dimensionality.

We can formally state this as follows:

Proposition 4.2 (Intrinsic dimension of generated image manifold) *Let dimension $\bar{d}$ and class $\mathbf{c} \in \mathbb{R}^c$ be fixed and let*

$$\mathbb{Z}^{\bar{d}} := \{(z_1, \dots, z_{\bar{d}}, 0 \dots, 0) | (z_1, \dots, z_{\bar{d}}, \dots, z_m) \sim \mathcal{D}_{noise}\} \subset \mathbb{R}^m$$

be the space of truncated noise vectors.
Then the set $G(\mathbb{Z}^{\bar{d}}, \mathbf{c})$ is almost everywhere locally a manifold with dimension less or equal to $\bar{d}$.

Proof Using that $\mathbb{Z}^{\bar{d}}$ is a manifold with dimension $\bar{d}$ and $G(\,\cdot\,, \mathbf{c})$ is the composition of Lipschitz continuous functions, the proposition follows from Sard's theorem [Sard, 1942]. $\square$

We use this approach to generate sets of images with different implicit dimension and can visually notice a difference: As we can see in figure 4.2, sets with low implicit dimension have a lot of similarity between samples. For example samples with $\bar{d} = 8$ always have a field as background and almost no variation in situation or perspective. In contrast samples with $\bar{d} = 128$ contain a large variety of settings and styles.

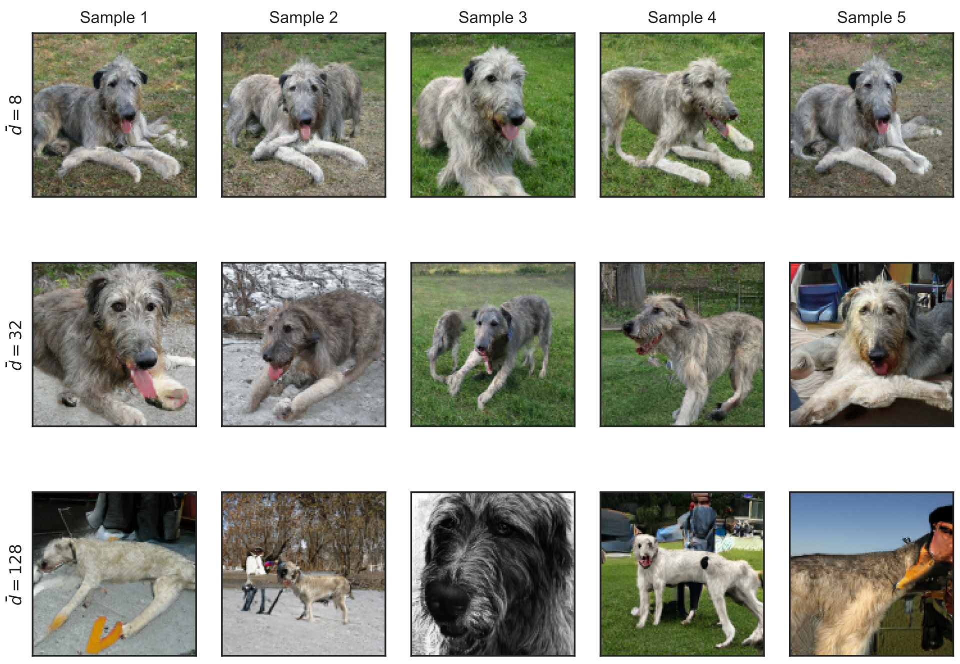


Figure 4.2: **Image samples from datasets with varying intrinsic dimension.** Images all belong to the same ImageNet class (irish wolfhound) and were generated by BigGAN128.

4.3 Results

In this section we will use the method detailed in the previous section to generate datasets with varying intrinsic dimension. We will then compare the performance and reset errors of networks trained on these datasets to investigate the effect of training difficulty on layer criticality.

4.3.1 Dataset

Using a BigGAN128 model we generate datasets of varying intrinsic dimension resembling the CIFAR10 dataset: First we select 10 ImageNet classes roughly corresponding to the 10 CIFAR10 classes (see appendix B for more details). Then for each each $\bar{d}$ in $\{32, 48, 64, 96, 128\}$ we generate 6000 images of each class with the respective intrinsic dimension. Next we combine the data such that we now have a dataset for each intrinsic dimension with each containing $60'000$ images evenly distributed across classes. And finally, to bring the datasets more in line with CIFAR10 and increase training speed, we scale the images down from 128×128 to 32×32 by applying MaxPooling

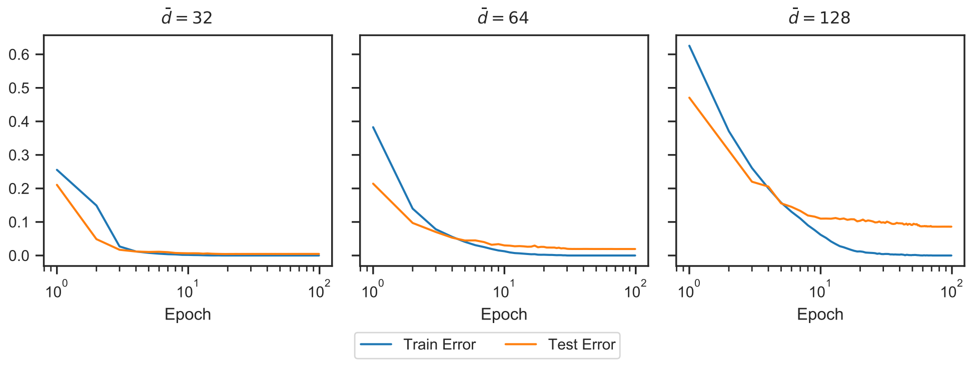


Figure 4.3: **Epoch-wise train and test error for models trained on datasets of varying intrinsic dimension.** Models are VGG11, trained for 100 epochs.

twice. Note that although this decreases dataset space dimension by a factor of 16 it is still much bigger than the intrinsic dimensions which are at most 128.

4.3.2 Training Difficulty and Criticality

We will now train VGG11 networks on the generated datasets of intrinsic dimension $\bar{d}$ in $\{32, 48, 64, 96, 128\}$. The results of the experiments turn out just as intuition might suggest: As intrinsic dimension increases the networks have more difficulty learning the data and previously ambient layers turn critical. While intuitively expected, this is a new result previously unknown in the literature. Moreover, this trend is a clear indication that a structured and balanced complexity allocation takes place during the training of deep learning architectures.

Training Difficulty

When we compare training difficulty across datasets with varying intrinsic dimension our results are in line with those reported by Pope et al. [2021]: As the intrinsic dimension gets higher the networks take longer to train and incur higher test error.

As we can see in figure 4.3 it takes roughly ten times the number of training epochs for a model trained on a dataset with $\bar{d} = 128$ to reach the interpolation threshold compared to a model with $\bar{d} = 32$. Furthermore, while at $\bar{d} = 32$ the model is able to achieve almost zero test error, a model trained at $\bar{d} = 128$ only reaches around 10%.

This signifies that intrinsic dimension is indeed a key factor in training difficulty and varying intrinsic dimension is a valid way of studying the effects of varying training difficulty on layer criticality.

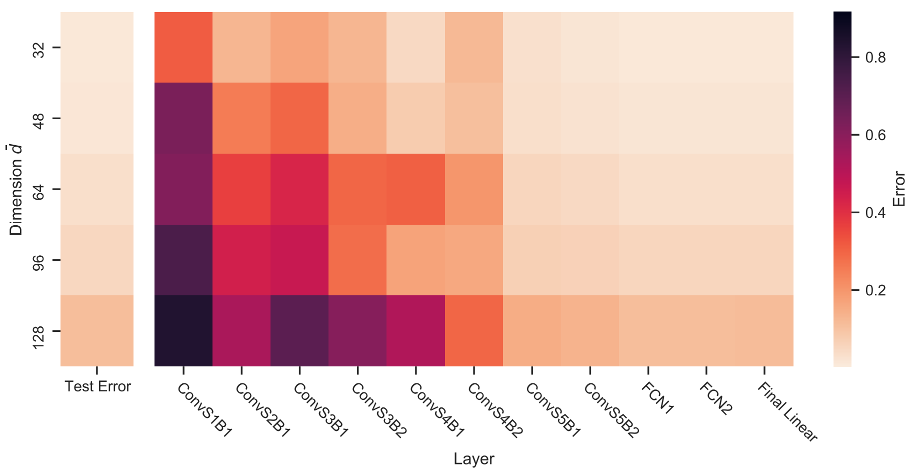


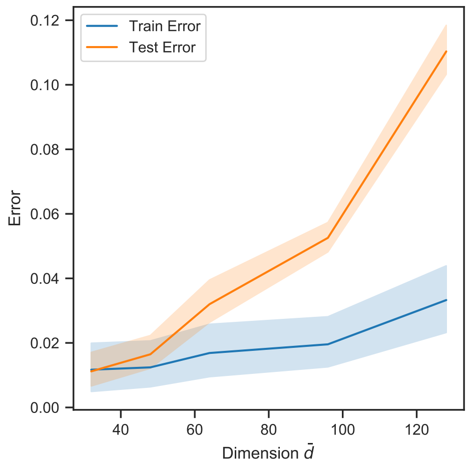
Figure 4.4: **Layer-wise reset error for datasets of varying intrinsic dimension.** Same layout as figure 3.3, but each row now corresponds to the test and reset errors of a model trained on the generated dataset with specified intrinsic dimension.

Layer Criticality

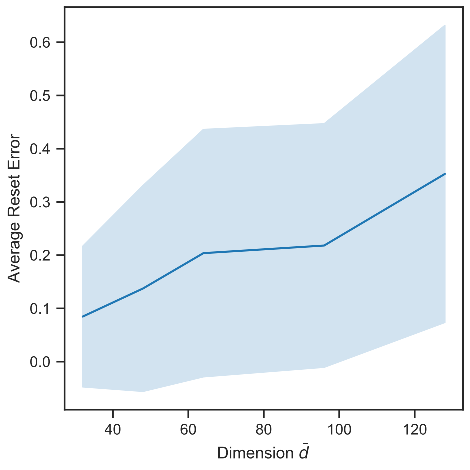
As we increase intrinsic dimension from 32 to 128 the reset error of layers does indeed react: While almost all layers are ambient at $\bar{d} = 32$, as the intrinsic dimension increases, we can see layers "activating" and switching from ambient to critical. But this effect is not distributed uniformly across the network: The layers close to the input quickly turn critical, while those further away are barely affected and stay ambient (see figure 4.4). This shows that the layers turning critical is not simply a byproduct of an overall increase in test error.

These results are significant as they suggest that the difficulty of the training task plays an important role in determining which layers turn critical and which turn ambient. We will elaborate on this further in chapter 5.

If we compare the mean and variance of the reset error across layers to the train and test error for varying dimensions (see figure 4.5), we can see that as in chapter 3 the mean reset error is correlated with the test error. Furthermore, we can see the variance in reset error increase as initially ambient layers turn critical.



(a) Test and train error



(b) Mean and variance of the reset error across layers

Figure 4.5: **Training and layer criticality metrics as a function of intrinsic dimension of the training dataset.**

Chapter 5

Conclusion

In this chapter we will first discuss the experimental results from the previous chapters to develop a unifying view. We will then outline possible avenues for further research into this topic.

5.1 Conjecture

The results described in chapters 3 and 4 allowed us to gain significant insight into the behavior of critical and ambient layers. We saw that a structured complexity allocation takes place during training which seems driven by width and reactive to training complexity. This leads us to informally conjecture the following:

Conjecture 2 (Behavior of Layer Criticality) *Training complexity is a key factor in determining* which *layers turn critical or ambient. Network width drives the differentiation into critical and ambient layers but has no influence over which layers are critical or ambient in the first place.*

Note that our conjecture is slightly different than the alternative explanation *As model capacity increases beyond the interpolation threshold, neural networks begin to self-regularize by reducing the number of critical layers.*
In line with this alternative explanation, as noted in chapter 3, we initially expected that increasing model width past the interpolation threshold would lead to previously critical layers turning fully ambient one by one. Doing so would mirror the behavior we observed in chapter 4 where increasing training difficulty would lead previously ambient layers to turn critical. But we *never* saw previously critical layers turn ambient past the interpolation threshold and many turned even more critical. Width thus seems to have a fundamentally different effect on layer criticality than training difficulty. This means layer criticality cannot be described by overcapacitation alone and this alternative explanation does not adequately capture its behavior.

In contrast, conjecture 2 is consistent with our observations and, as we will see in the next section, also makes sense considering related results.

5.2 Discussion

We can gain an intuition for the idea that width has no influence over which layers turn critical or ambient by considering the following: While in theory networks with sufficient width and one hidden layer are universal function approximators, in practice a certain network *depth* enables models to develop higher level input representations and simplifies the training task [Arora et al., 2018, Lederer, 2021], allowing deep networks to solve problems which would be infeasible with just one hidden layer. This suggests that a certain amount of "active" layers is required for a given training task, which increasing width cannot change.

Additionally, layers which can be reset to their initial state decrease model complexity, as they can be viewed as just a nonlinear transformation of the output space which does not change during training. The separation into fully ambient and fully critical layers is therefore a type of self-regularization in neural networks. And since width has been observed to be a self-regularizing force in neural networks [Belkin et al., 2019, Mei and Montanari, 2019], it is not surprising for width to be a key factor in this phenomenon.

Furthermore, it is interesting that for double descent in width, the second drop in test error coincides with the separation into fully critical and ambient layers. As previously mentioned it is a commonly held belief that double descent is caused by some kind of self-regularization which causes the network to choose an interpolating solution that generalizes well. Considering the development of critical and ambient layers can be viewed as a kind of self-regularization, one might suspect a possible connection between layer criticality and double descent. However, our results are too limited to suggest such a connection exists with confidence.

Note that although training complexity is a key factor in determining layer criticality, it is not the only one. Architectural choices such as the number and type of layers also play a significant role. As an example we can consider the case of residual networks: Zhang et al. [2019] found that for these network architectures, instead of the layers closer to the input turning critical and those further away turning ambient, we see that the first layers of the residual blocks turn critical, while others are ambient. This makes sense considering the findings of Veit et al. [2016], which show that residual networks behave as ensembles of smaller, shallow networks.

In conclusion, we believe that layer criticality plays an important role in understanding the generalization behavior of deep neural networks and our results allow us to significantly advance our understanding of this phenomenon.

5.3 Further Directions

There are many promising directions one could further pursue that were unfortunately beyond the scope of this thesis:

As mentioned in the previous section, network architecture also plays an important role in determining layer criticality and one could try to identify *which* layers turn critical under what circumstances.

Another interesting direction would be to further investigate the possible link between double descent and layer criticality to determine if and how layer criticality impacts generalization: Zhang et al. [2019] used layer criticality to improve existing generalization bounds and as we saw increasing width leads to a sharper separation into critical and ambient layers. This could possibly be combined into generalization bounds capable of explaining the double descent in width.

And finally, one could try to understand *how* width causes the separation into critical and ambient layers. To that end one could consider layer criticality in the infinite width limit or develop synthetic training tasks to isolate this behavior.

5.4 Acknowledgments

We thank Professor Fanny Yang for suggesting this interesting topic and overseeing this thesis. We thank Professor Nicolai Meinshausen for assisting in the organization of this thesis.

We also thank Nicolò Ruggeri for many useful discussions and suggestions. Without his guidance this thesis wouldn't have been possible.

Appendix A

Training and Model Parameters

A.1 Model Architectures

All models except those trained on MNIST are of VGG11 architecture developed by Simonyan and Zisserman [2015]. VGG11 consists of convolutional layers of increasing width interspersed with MaxPooling layers organized in blocks, followed by two hidden layers of width 4096. For an input of 32×32 images this architecture has $28'144'010$ parameters.

Models trained on MNIST use a reduced VGG11 architecture with the final convolutional block removed

A.2 Training Parameters

In line with Zhang et al. [2019] we use SGD with momentum and a base learning rate of 0.01. We do not use batch normalization or any type of external regularization since we want to keep the number of factors influencing layer criticality as small as possible. Models use a batch size of 128 and were trained for 100 epochs. All results reported in this thesis are averaged over at least five trials to reduce noise.

In all experiments, except those concerning double descent in varying training time (section 3.3), a piecewise constant learning rate decay is used: For the double descent in width experiments (section 3.2) we use the same learning rate decay as Zhang et al. [2019]: learning rate is scaled down by a factor of 0.2 after epochs 30, 60 and 90. In the case of varying intrinsic dimension (section 4.3.2) we are worried about the learning rate decaying too quickly and only scale it down once by 0.2 at epoch 70.

Appendix B

GANs and Dataset Generation

B.1 GANs In-depth

Here we will give a more in-depth description of the learning process of GANs:

As previously mentioned GANs consist of two networks: A generator G tasked with generating images mimicking those in the original dataset and a discriminator D tasked with distinguishing between generated and real images.

Formally the training task can be described as follows: Let

$$D_\theta : \mathbb{R}^k \to [0,1]$$

be the discriminator network with parameters θ mapping images in the input space $\mathbb{R}^k$ to $[0,1]$ representing probability of the input image being real. Note that for the sake simplicity and uniform notation we will represent images as vectors in $\mathbb{R}^k$ Conversely let

$$G_\varphi : \mathbb{R}^m \to \mathbb{R}^k$$

be the generator network with parameters φ, generating images by mapping a noise vector in $\mathbb{R}^m$ to the image space $\mathbb{R}^k$.

With samples from the data distribution $\mathbf{x} \in \mathbb{R}^k \sim \mathcal{D}_{data}$ and samples from the noise distribution $\mathbf{z} \in \mathbb{R}^m \sim \mathcal{D}_{noise}$ we now simultaneously train the discriminator and the generator networks: D_θ is trained to maximize the probability of assigning correct labels to training samples and samples generated by G:

$$\max_\theta \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_{data}}[\log D_\theta(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim\mathcal{D}_{noise}}[\log(1 - D_\theta(G_\varphi(\mathbf{z})))]$$

Conversely G_φ is trained to minimize the probability of D_θ classifying a generated image as fake:

$$\min_\varphi \mathbb{E}_{\mathbf{z}\sim\mathcal{D}_{noise}}[\log(1 - D_\theta(G_\varphi(\mathbf{z})))]$$

The specific procedure for training both objectives simultaneously is outlined in algorithm 1.

Algorithm 1: Training procedure for generative adversarial nets. The number of discriminator updates k per training iteration is a hyperparameter.

```
for number of training iterations do
    Update the discriminator:
    for k steps do
        Sample a batch of $m$ noise samples $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)} \sim \mathcal{D}_{noise}$
        Sample a batch of $m$ data samples $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)} \sim \mathcal{D}_{data}$
        Perform a stochastic gradient descent update on the discriminator
          with loss
            $-\frac{1}{m}\sum_{i=1}^{m}\left[\log D_\theta(\mathbf{x}^{(i)}) + \log(1 - D_\theta(G_\varphi(\mathbf{z}^{(i)})))\right]$
    Update the generator:
    Sample a batch of $m$ noise samples $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)} \sim \mathcal{D}_{noise}$
    Perform a stochastic gradient descent update on the generator with
      loss
        $\frac{1}{m}\sum_{i=1}^{m}\log(1 - D_\theta(G_\varphi(\mathbf{z}^{(i)})))$
```

B.2 Dataset Generation

As mentioned in chapter 4, the datasets of varying intrinsic dimension were created by selecting 10 ImageNet classes which roughly correspond to the 10 classes of Cifar10. Figure B.1 shows samples (which have already been scaled down to 32×32 resolution) drawn from the generated dataset of intrinsic dimension 64 labeled with both the CIFAR10 class name and the corresponding ImageNet class name.

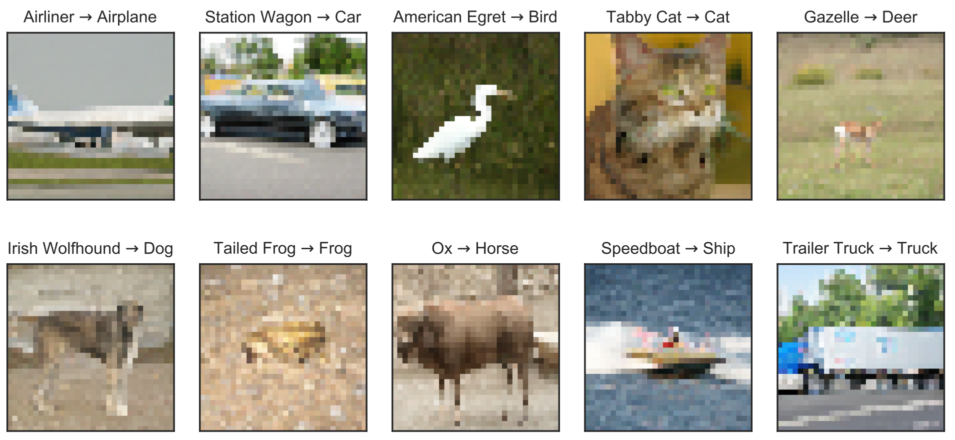


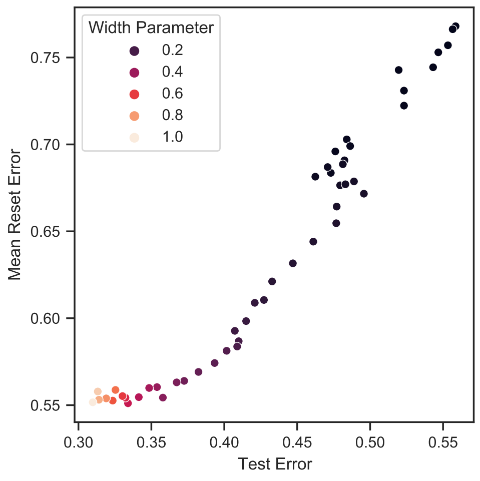
Figure B.1: **Sample images of each class of the generated datasets. Labeled by their ImageNet class and the CIFAR10 class they are supposed to correspond to.** The dataset has intrinsic dimension $\bar{d} = 64$ and the images have been downscaled from 128×128 to 32×32 by applying MaxPooling twice.

Appendix C

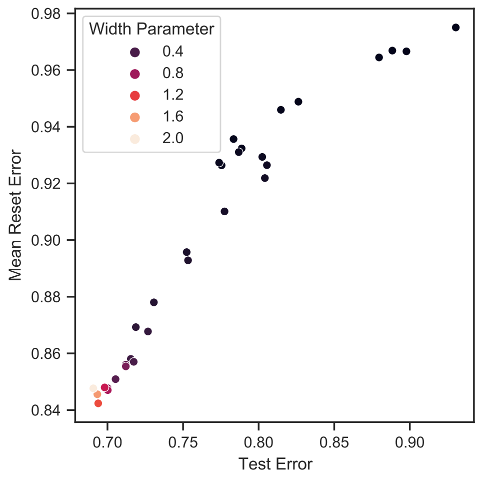
Additional Plots

C.1 Correlation of Test Error and Mean Reset Error

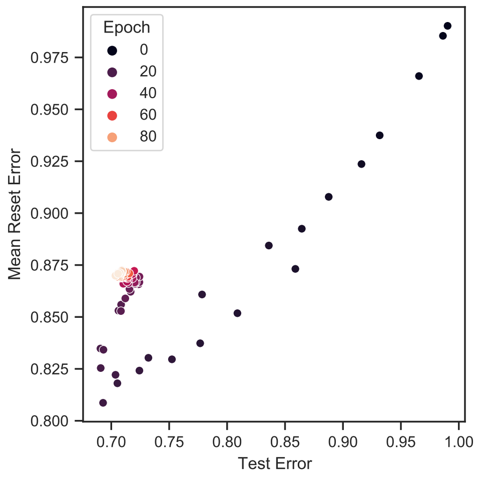
In figure C.1 we can see that, as previously mentioned, test error does correlate with mean reset error under some circumstances. We see that this correlation is strongest when model capacity is small (i.e. EMC $< n$ which means we are in the underparameterized regime) and starts to break down as model capacity increases (i.e. EMC $> n$ which means we are in the overparameterized regime).



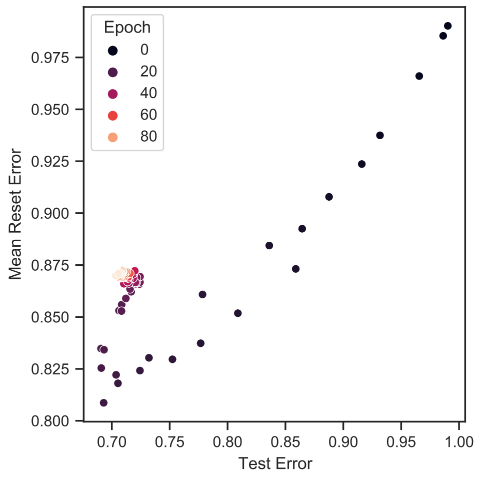
(a) CIFAR10 and varying width



(b) CIFAR100 and varying width



(c) CIFAR10 and varying training epochs



(d) CIFAR100 and varying training epochs

Figure C.1: **Scatterplots of test error and mean reset error across datasets and observed hyperparameter.** Each dot denotes the test error and the mean reset error for a model trained with a specific width parameter / trained for a specific number of epochs. Dots which are colored brighter correspond to higher width parameter / more training epochs. Models are of VGG11 architecture and are trained on CIFAR10 and CIFAR100 with 15% label noise added.

C.2 Layer-wise criticality for varying training epochs

Figure C.2 allows us to gain a more in-depth look at the development of layer criticality in the case of varying training epochs. We can again see the separation into critical and ambient layers as shown in figure 3.8. We can also see how most of the change in layer criticality happens *before* the model reaches the interpolation threshold. This is especially clear if we compare this figure to the corresponding figure 3.6 for the case of varying width.

The separation into critical and ambient layers doesn't continue past the interpolation threshold, which further solidifies the idea that increasing training epochs doesn't cause layer to separate further, but layers can be prevented from separating if the model is trained for too few epochs.

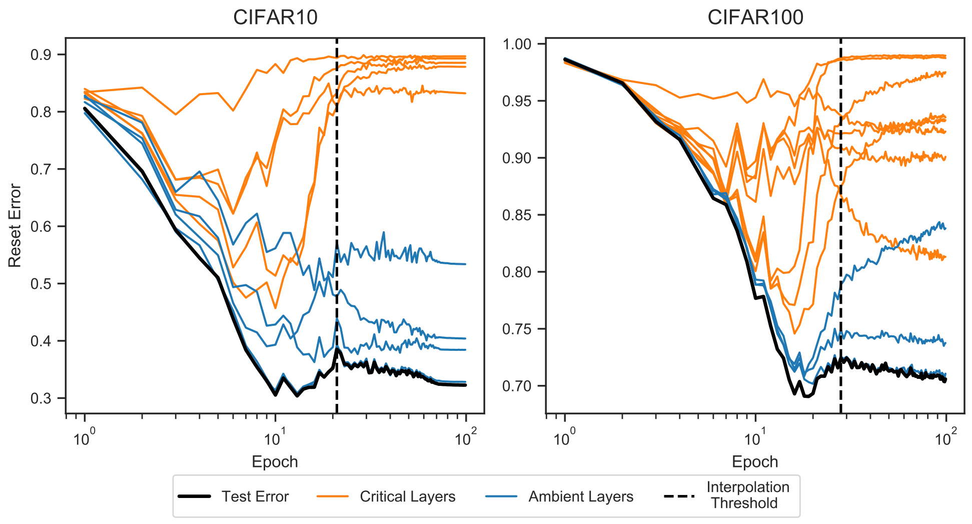


Figure C.2: **Layer-wise criticality and test error as a function of training epochs.** Layout it the same as in figure 3.6, but instead of model width we vary the number of training epochs. The layers colored critical and the layers colored ambient are also the same as in 3.6 to allow a comparison of the two figures.

Bibliography

Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. *PMLR*, 2018.

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine learning practice and the bias-variance trade-off. *PNAS*, 2019.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *ICLR*, 2019.

Niladri Chatterji, Behnam Neyshabur, and Hanie Sedghi. The intriguing role of module criticality in the generalization of deep networks. *ICLR*, 2020.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. *CVPR*, 2009.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *NeurIPS*, 2014.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *ICLR*, 2016.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009. URL `https://www.cs.toronto.edu/~kriz/cifar.html`.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010. URL `http://yann.lecun.com/exdb/mnist/`.

Johannes Lederer. Optimization landscapes of wide deep neural networks are benign. *arXiv preprint arXiv:2010.00885*, 2021.

Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and double descent curve. *arXiv preprint arXiv:1908.05355*, 2019.

Vaishnavh Nagarajan and J. Zico Kolter. Uniform convergence may be unable to explain generalization in deep learning. *NeurIPS*, 2019.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *ICLR*, 2019.

Hariharan Narayanan and Sanjoy Mitter. Sample complexity of testing the manifold hypothesis. *NeurIPS*, 2010.

Phil Pope, Chen Zhu, Ahmed Abdelkader, Micah Goldblum, and Tom Goldstein. The intrinsic dimension of images and its impact on learning. *ICLR*, 2021.

Arthur Sard. *The measure of the critical values of differentiable maps*, volume 48. American Mathematical Society, 1942.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

Andreas Veit, Michael Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. *NeurIPS*, 2016.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *ICLR*, 2017.

Chiyuan Zhang, Samy Bengio, and Yoram Singer. Are all layers created equal? *ICML*, 2019.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.