2024-04-29

An Investigation of the Interactions of Gradient Coherence and Network Pruning in Neural Networks

Zachary Yauney
Brigham Young University

An Investigation of the Interactions of Gradient Coherence and Network Pruning in Neural Networks

Zachary Yauney

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Tyler Jarvis, Chair
Jared Whitehead
David Wingate

Department of Mathematics

Brigham Young University

ABSTRACT

An Investigation of the Interactions of Gradient Coherence and Network Pruning in Neural Networks

Zachary Yauney
Department of Mathematics, BYU
Master of Science

We investigate the coherent gradient hypothesis and show that the coherence measurements are different on real and random data regardless of the network's initialization. We introduce "diffs," an attempt at an element-wise approximation at coherence, and investigate their properties. We study how coherence is affected by increasing the width of simple fully-connected networks. We then prune those fully-connected networks and find that sparse networks outperform dense networks with the same number of nonzero parameters. In addition, we show that it is possible to increase the performance of a sparse network by scaling the size of the dense parent network it is derived from. Finally we apply our pruning methods to ResNet50 and ViT and find that diff-based pruning can be competitive with other methods.

Keywords: Deep learning, neural networks, generalization, gradient coherence, sparsity, pruning

Acknowledgements

Thanks to Dr. Jarvis, for letting me follow an interesting lead even though it meant more work. Also thanks to Michael Nelson, whose work I originally set out to build on. It was in reading his thesis that I first came across the "Coherent Gradients" hypothesis, and in trying to replicate his experiments that I had the idea for tracking diffs.

In addition, thanks to Dr. Whitehead and Dr. Wingate for their patience while I took an entire additional year to refine the ideas originally presented in a hasty attempt to submit something before the end of the semester. It is my hope that the additional time has resulted in a commensurately better final product.

Contents

Introduction

While the study of deep learning has experienced a meteoric rise in the last decade, much of the underlying dynamics of network training remains poorly understood. Often new architectures are designed via trial and error rather than a priori reasoning. This is mostly due to a lack of tools and theorems to support such reasoning. At the beginning of this project we set out to contribute to the development of the fundamental theory of deep learning, to aid in the design of better architectures and training protocols and perhaps even understand what causes deep learning to be so astonishingly effective. As has often been the case in science, empirical study is required in order to provide clues for theory to follow. To this end, this thesis presents a series of empirical studies aimed at developing some effective footholds for the future continued empirical and theoretical development of our collective understanding of the fundamentals of deep learning.

As is typical in research we arrived at the ideas in this paper by following a somewhat circuitous path. We began with an intent to study properties of networks during training that might be correlated with superior performance, following Nelson (2022). This led us to Chatterjee and Zielinski (2022) and the introduction of the coherent gradients hypothesis. A misunderstanding about how computationally expensive it would be to calculate coherence led to the creation of "diffs," an element-wise approximation of coherence. This led to an interest in the possibility of using such a metric for network pruning, which resulted in a study of a few of the properties of network pruning. The result is a series of insights about the intersection of generalization, gradient coherence, and network pruning. While we certainly present some interesting findings, the implications of these findings are somewhat unclear. Many opportunities for further research remain.

Chapter 1. Generalization and Gradient Coherence

We begin this chapter with a study of what has already been learned about the generalization problem in deep learning. Then we detail an idea known as the "coherent gradients hypothesis" and investigate their position in the current literature. We conclude with an outline for the rest of the paper.

1.1 Literature Review on Generalization

One of the chief questions in modern neural network research is why neural networks generalize so well despite being wildly overparametrized. Traditional machine learning theory holds that generalization stems from a limiting of the hypothesis class to a small enough set that overfitting is controlled. Neural networks, however, clearly violate this notion. There are three main ways that they do so.

Neyshabur et al. (2015) showed that increasing the number of hidden units in a network can actually improve network generalization instead of hindering it. In this simple experiment the researchers looked at fully-connected networks with a single hidden layer of increasing size trained on MNIST and CIFAR-10. They found that test error actually decreased consistently with the size of the hidden layer. This is unexpected because we see increased capacity resulting not in overfitting but in improved performance on the test set.

Zhang et al. (2017) provides even more shocking continuation of this idea with an experiment that showed that a convolutional network was capable of memorizing a randomly labeled dataset (clear overfitting) and yet also generalizing well when trained on properly labeled data. This means that when training on real data the model has the capacity to totally overfit, and yet it doesn't. The result is unaffected by explicit regularization and even happens when the training data is just Gaussian noise. (Zhang et al., 2021) This result means we must rethink our understanding of generalization, because limiting capacity clearly

isn't necessary to enable generalization.

Finally, it was shown that a 2-layer fully-connected network has perfect finite sample expressivity as soon as the number of parameters exceeds the number of data points. (Zhang et al., 2021) In practice networks are far larger than this. This theoretical result solidifies the empirical evidence that something other than limited network capacity must be responsible for neural network generalization. Historically there have been several hypotheses to explain generalization, with varying degrees of promise.

Classical machine learning holds that VC dimension is the key to controlling generalization. Baum and Haussler (1989) showed VC dimension inflates rapidly with parameter count and thus we should need far more training data to generalize effectively. This classical view of generalization is also directly contradicted by the previously mentioned experimental evidence. Nagarajan and Kolter (2021) also demonstrates that the notion of uniform convergence is unsuitable because its estimation of the generalization gap can sometimes increase with more training data when in practice more training data improves generalization. Uniform convergence is the idea that it is possible to sufficiently restrict an algorithm's hypothesis class to limit the maximum generalization error for a classifier trained with that algorithm on any sampling of the target data distribution. The authors provide a theorem showing that with the right setup, "even the tightest uniform convergence bound is nearly vacuous despite good generalization." Because generalization is exhibited even when uniform convergence does not guarantee it will be, there must be another explanation for why over-parametrized networks generalize.

A more sophisticated technique is to compute norms on the matrices that compose the network parameters and try to establish a relationship between the norms and the generalization of the network. Bartlett (1998) relates the $\mathcal{L}_1$ norm of the parameters to the misclassification rate, arguing that smaller parameter values might lead to better generalization outcomes. Neyshabur et al. (2015) used an analogy to matrix factorization to suggest the use of other matrix norms such as the trace norm. They suggest that the learning

process works because optimization methods bias the learning process towards low-norm models, even if the parameter count is high.

Margin bounds are another proposed generalization bound. Glasgow et al. (2023) attempts to explain how generalization can occur in regimes where uniform convergence fails. They demonstrate the existence of a class of solutions that are near-max-margin, where uniform convergence fails but test loss can be driven arbitrarily low given enough training data. This is a promising line of inquiry but is not studied in this paper.

The final measure to mention in this review is sharpness. It has been proposed that sharp minima, that is minima where the surrounding gradients are large, generalize less well than flat minima. It was introduced Hochreiter and Schmidhuber (1997) when the authors described an algorithm for finding "flat minima" that were meant to generalize well. Sharpness was again linked to the generalization mystery in Keskar et al. (2017) as part of their investigation into why large batch sizes seem to inhibit generalization. They found that the large-batch methods "tend to converge to sharp minimizers of the training and testing functions," explaining why the large-batch methods seem to generalize poorly. Neyshabur et al. (2017) established theoretical bounds on sharpness for simple feedforward networks with RELU activations. A useful feature of these bounds is that they do not increase exponentially with network depth, as other measures do.

1.2 Introducing Gradient Coherence

Chatterjee and Zielinski (2022) takes on the generalization question with a novel approach centered on what they term "gradient coherence." Informally, gradient coherence is a measure of the degree to which the gradients of different samples in the training data help the network better classify other samples in the training data. Formally, with samples z from a dataset $\mathcal{D}$ and per-sample loss function gradients g_z, the equation that they give for coherence (α) is

$$\alpha = \frac{||\mathbb{E}_{z\sim\mathcal{D}}[g_z]||^2}{\mathbb{E}_{z\sim\mathcal{D}}[||g_z||^2]}.$$

A good way to think about coherence is that it is a measure of how strong of a generalizable patterns the network is learning during training. When training on real data where patterns abound, coherence is high. When training on random data that provably has no patterns, coherence is low. Chatterjee and Zielinski (2022) believe that there are essentially two regimes of neural network learning. In the first regime, generalizable patterns are learned. Then later, the "hard" samples that do not fit into any of generalizable patterns are memorized. This is supported by the fact that some samples are consistently learned earlier in training than others. These are the "easy" samples that fall within the generalized learning. The "hard" samples learned later are the ones that need to be memorized.

Two important features of gradient coherence are that it is independent of the weights or norms of the network parameters, and that it is influenced by the dataset being trained on. We believe that it is likely that a good measure of generalization will have to involve the dataset being trained on.

Figure 1.1 summarizes the results of their experiments, showing that coherence is in fact significantly different when calculated for a network training from initialization on real data as opposed to a network training on randomly generated data. For their experiments they used a ResNet50 training on ImageNet with SGD and a learning rate of 0.5. The logic behind this experiment is that generalization does occur when training on real data, and cannot occur when training on random data, so differences in the network's behavior when training on these datasets might be able to be attributed to the way the network behaves when when generalization is occurring as opposed to when memorization is occurring.

To summarize, the Coherent Gradient Hypothesis states that generalization occurs when gradients from different samples tend to point in the same direction. Experiments suggest that gradient coherence (α) is correlated with the network learning a generalizable representation of the data, as α is larger when training on real as opposed to random data.

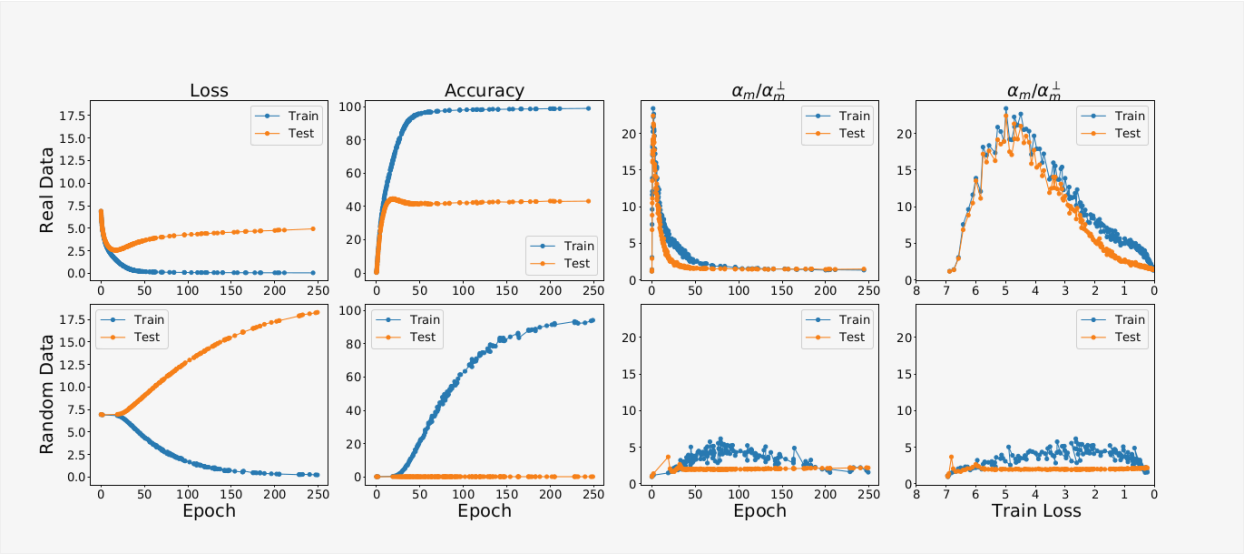

Figure 1.1: The original figure from Chatterjee & Zielinski. The top row provides results for training ResNet50 on ImageNet data. Coherence sees a large peak early in training. The second row gives results for training ResNet50 on Gaussian noise in the shape of ImageNet images. $\alpha_m/\alpha_m^\perp$ is their measure of coherence multiplied by the size of the dataset, a scaled version of the measure introduced previously. Column 3 depicts the evolution of coherence by epoch, and column 4 shows how coherence changes as the train loss decreases. Although training loss goes to near zero, coherence remains low throughout training. There is a much smaller spike in coherence coinciding with the epochs where the network is memorizing the random data.

1.3 Gradient Coherence in the Literature

Despite the interesting claims that Chatterjee and Zielinski make about gradient coherence, not much has been written about investigations into this phenomenon. Most of the references to their work simply reference in passing the notion that coherent gradients are good or that easy samples are learned more quickly than hard samples. We will, however, relate one paper that directly touches on the subject of gradient coherence, perhaps even providing some explanation for what causes gradient coherence.

Sabanayagam et al. (2023) examines the relationship between a network's Hessian and the decision boundary. The decision boundary is important because a simple decision boundary is a known indicator of generalization. They first note that when the network is at a minimum, the spectrum of the Hessian separates into a multitude of values clustered near zero along with a few outliers. They then note that the number of outliers is generally approximately equal to the number of classes in the dataset the network is trained on. In addition, the gradient information "resides in a small subspace spanned by the Hessian top eigenvectors." These are all observations from previous works.

Their contribution is to connect the outliers in the spectrum of the Hessian to the simplicity of the decision boundary. Through numerical analysis they show that not only is the gradient information mostly spanned by these outlier eigenvectors, so is the decision boundary. Each of the eigenvectors encodes its own separate section of the decision boundary. They hypothesize that gradience coherence emerges from a simple decision boundary, which would explain why high coherence is associated with high generalization.

1.4 Outline

Now that we have covered the background on generalization and its relation to gradient coherence, we will attempt to explore and exploit these coherent gradients in the context of network pruning. The outline of this thesis is as follows:

1. We begin by reproducing the results of the original gradient coherence paper, as well as performing an additional experiment to show that coherence is in fact a special property of gradients trained on real data.

2. We introduce a measurement we refer to as "diffs" which function as an element-wise approximation of coherence within a network.

3. We examine how coherence varies with layer width in feedforward fully-connected networks.

4. We introduce diff-based pruning using diffs, and investigate properties of pruning in the context on fully-connected networks.

5. We apply our new diff-based pruning techniques to large models such as a vision transformer and ResNet50

Chapter 2. Reproducing The Coherent Gradients Experiment

Before diving deep into an exploration of the properties of gradient coherence, we first validate the original experiments. In addition to reproducing the original results, we will perform other experiments to clarify whether coherence is a merely a product of movement in weight space or actually an indicator of coherence.

2.1 Reproduction of Original Results on CIFAR-10

Here we introduce two large models and associated hyperparameters that we will use throughout this paper. The first is ResNet50, the same network as used in the original coherent gradients paper. We follow their protocol and use standard stochastic gradient descent with a constant learning rate of 0.1. We train on real data for 300 epochs, and when training

on random data we train for 1000 epochs, which is enough for the networks to memorize Gaussian data.

The other network that we use is the Vision Transformer (ViT). We use a patch size of 4, a depth of 6, 8 QKV heads, and a fully-connected network with a width of 512. We also use no dropout or other explicit regularization. This model is trained using Adam with a learning rate of 1e$-$4. ViT models are trained for only 200 epochs on real data, and 1000 epochs on random data.

When we say that a network is trained on "real" data, that means the CIFAR-10 dataset. "Gaussian" or "random" data is images in the shape of CIFAR-10 images, but with each pixel being sampled from a Gaussian distribution, meaning the image is entirely random noise. The Gaussian data is given one of the 10 CIFAR-10 labels.

Our reported calculation of coherence differs slightly from that reported by Chatterjee and Zielinski (2022) because they scale their measurement by the size of the dataset and I do not. They use this scaling because for a dataset with m samples, α would take on the value $\frac{1}{m}$ in the case that all gradients were all pairwise orthogonal, a value they call the "orthogonal limit." They use this scaling to distinguish their measurement of α on a sample of the underlying distribution $\mathcal{D}$ from the value it would have if it could be measured on the true distribution. Because I am concerned more with comparison of α between architectures and datasets, I simply calculate the ratio between the norm of the expected value of the gradient and expected value of the norm of the gradient. This means that the range of my measurement is $[0, 1]$. One of my aims is to report a coherence score that is comparable across different networks and different datasets, allowing for easier comparison. In addition, this lack of scaling prevents confusion between my coherence measurements and theirs, as they trained on ImageNet and not CIFAR-10 so our dataset sizes are different and we would obtain different values for $\alpha/\alpha_m^{\perp}$.

Figures 2.1 and 2.2 detail the results of training these models on real and Gaussian data. The ResNet is able to memorize the Gaussian training data, as in Chatterjee and Zielinski's

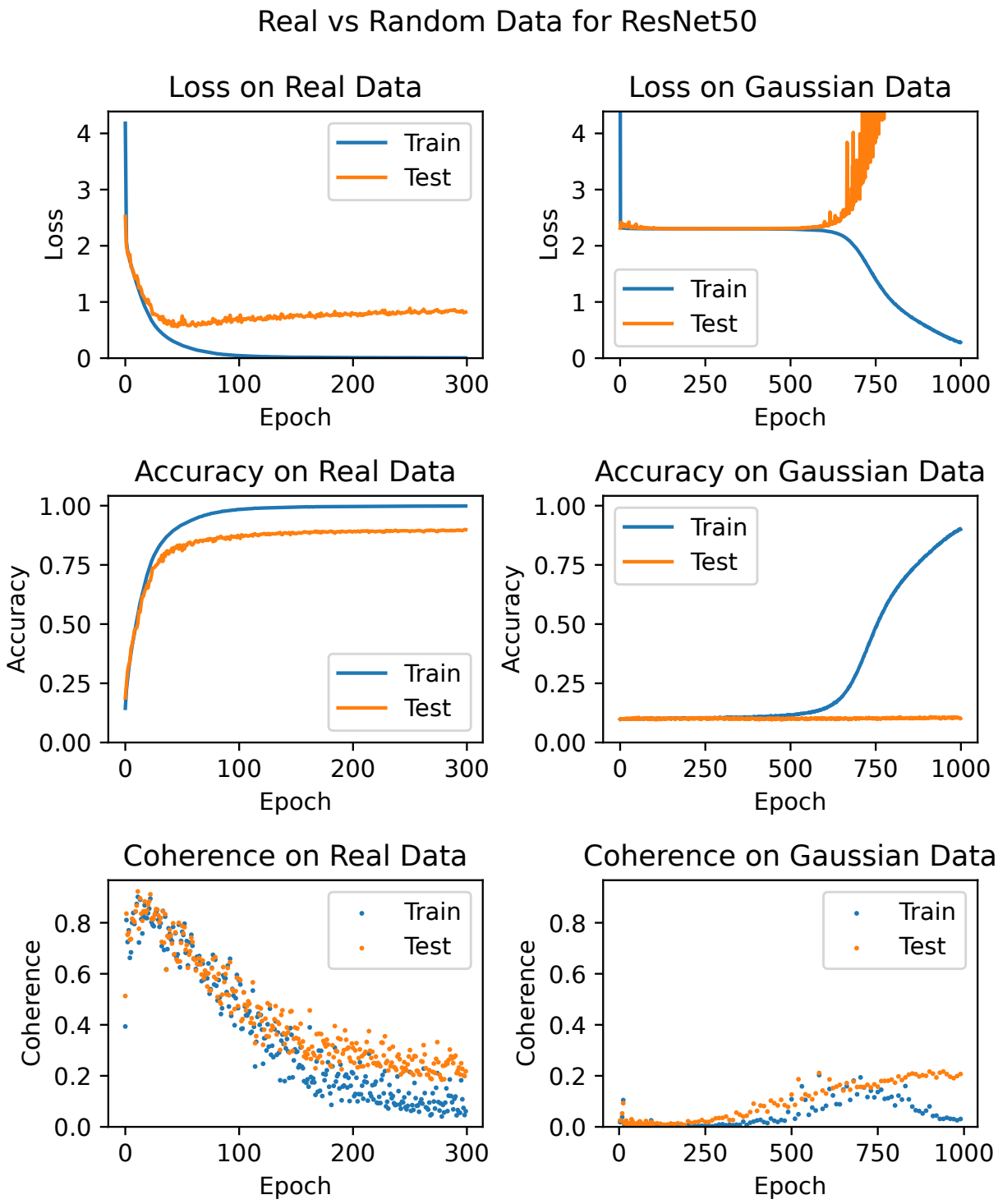

Figure 2.1: Results from ResNet50 trained on both real and Gaussian data. The network is able to achieve perfect training loss even on the Gaussian data, although the test loss is still high on the Gaussian data. Notice that coherence obtains a larger maximum when training on real data, as is predicted by the coherent gradients hypothesis. Compare to Figure 1.1 which illustrates results for the same experiment.

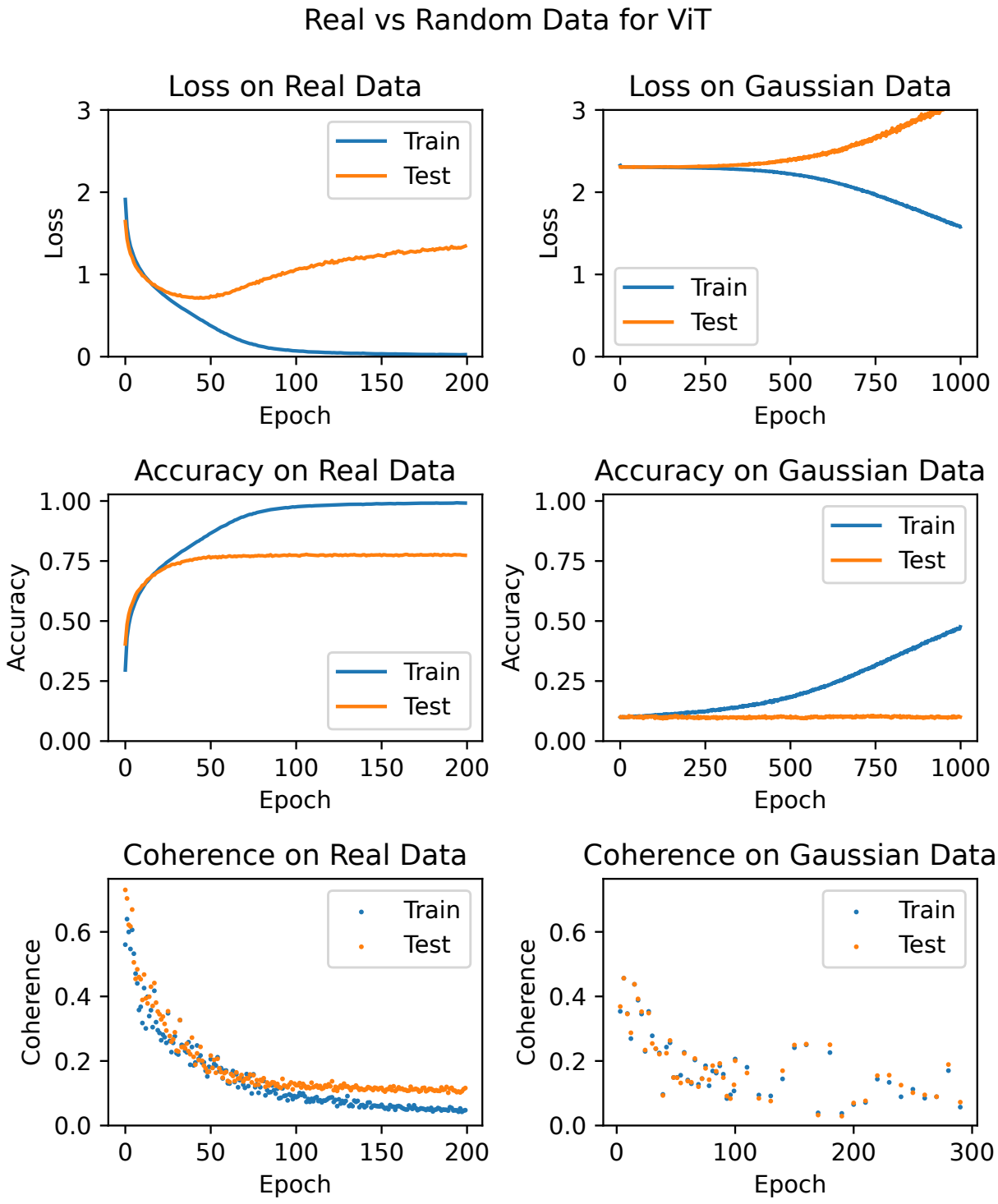

Figure 2.2: Results from ViT trained on both real and Gaussian data. Unlike with ResNet50, the ViT is unable to memorize the Gaussian training data. However, the early epochs still show significantly higher coherence when trained on real data vs Gaussian data, which remains in line with Figure 1.1.

original experiment, and the ViT model was able to memorize some of the Gaussian data. Also following their experiments, the coherence measurement on real data was noticeably higher than the coherence on Gaussian data. This establishes a correlation between high coherence and generalizable learning. Our next analysis will be aimed at attempting to further establish causality for this relationship as opposed to just correlation.

It should be noted that while coherence is higher in the beginning when trained on real data, in later epochs it is higher for the Gaussian data. Chatterjee and Zielinski address this in their paper, explaining that as the network learns the gradients flatten out and the precise direction they point become more stochastic near the minima found by the optimizer. This means that coherence will naturally rise to a peak as learning progresses and then it will fall off when no more learning is occurring. Because training on Gaussian data takes longer, coherence is slower to drop off entirely. One takeaway of this analysis is that if we want to examine the properties of the coherence gradients, we should be examining the model at the specific epochs where coherence is highest. This insight will factor into our future experiments.

2.2 Establishing Causation Between Coherence and Generalization

In seeking to establish causation between coherence and generalization, we seek to rule out alternate explanations for their correlation. One such explanation is that on each dataset, there is a particular region of weight space that contains the optimizers, and high coherence occurs when all the weights move from their initialization point into this optimizer region. It would then be possible that the distance between initialization and the real data optimizer is larger than the distance between initialization and the Gaussian data optimizer, leading the real data optimizer to exert a stronger pull on the weights and generate a stronger coherence effect. We term this idea that faraway optimizers have a stronger pull on weights and thus induce higher coherence the "gravity explanation" for coherence.

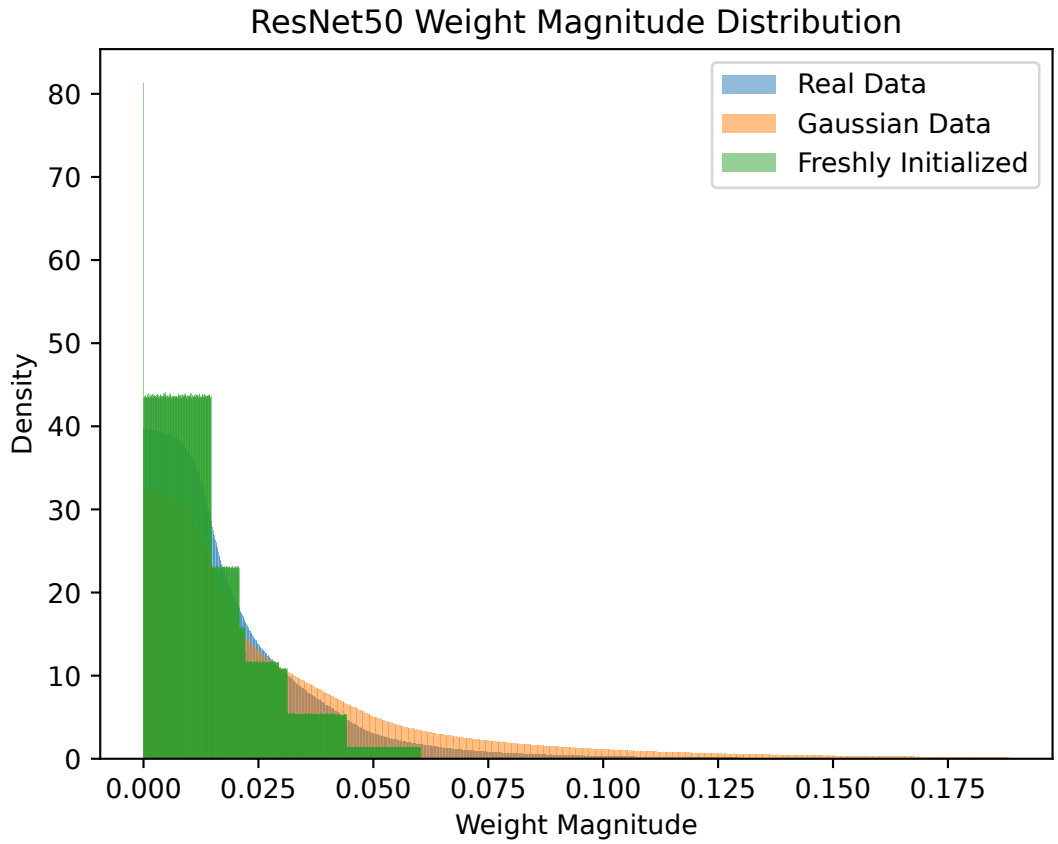

Figure 2.3: Distribution of bottom 99% of weight magnitudes for ResNet50 trained on real and Gaussian data, as well as at the network's random initialization. The fact that the weight magnitudes are very similar between networks trained on real and Gaussian data suggests that a difference in how the weight magnitudes evolve is not responsible for the difference in measured coherence.

To rule out this possibility, we first examine the distribution of weight magnitudes in the trained networks compared to newly initialized networks. If the final weights for Gaussian training are indeed closer to the initialization weights than the weights trained on real data, that might point towards the gravity explanation being a reality.

Figure 2.3 shows the weight magnitude distributions for ResNet at initialization as well as after training on real and Gaussian data. The top 1% of weights are discarded because they are so much larger than the other weights as to destroy the histogram. Because coherence is about the behavior of the majority of weights, this shouldn't affect the validity of the plots for ruling out the gravity explanation for coherence. Interestingly, we find that there are almost

no differences between the distributions of the majority of the weights for networks trained on both real and Gaussian data. This strengthens the case for a causal relationship between coherence and generalization, as differences in the distribution of weight magnitudes do not seem to be a determining factor in measured coherence. Ruling out this explanation gives us more confidence in the explanation that coherence is genuinely linked to generalization.

We ran one more experiment to rule out the gravity explanation for coherence. We initialized a ResNet50 network with weights derived from fully training on real data, and then train that network on Gaussian data. If coherence is lower on Gaussian data because the final weights are closer to the initialization point than the final weights for a network trained on real data are to the initialization point, then this network initialized with trained weights should exhibit high coherence when training on Gaussian data. We would expect to see a similar level of coherence because the weights are all migrating.

Figure 2.4 shows that this is not the case. Coherence is significantly higher for the network that is randomly initialized and trained on real data. The network trained on Gaussian data does see a small spike in coherence between epochs 100 and 200. However, that spike is not nearly as strong as the spike seen when the network is learning the real data. This plot is evidence for a causal relationship between high coherence and generalization.

Now that we have demonstrated the value of coherence as a tool for investigating generalization, we will use it to derive a new tool that allows us to utilize the notion of coherence in the domain of network pruning.

Chapter 3. Diffs: An Elementwise Measure of Coherence

We begin this chapter by defining diffs, which are intended to serve as an element-wise approximation of gradient coherence. After defining diffs, we examine the distribution of the diffs that result from training. These distributions support the link between diffs and coherence. Finally, we examine how the values of high and low diffs parameters evolve over

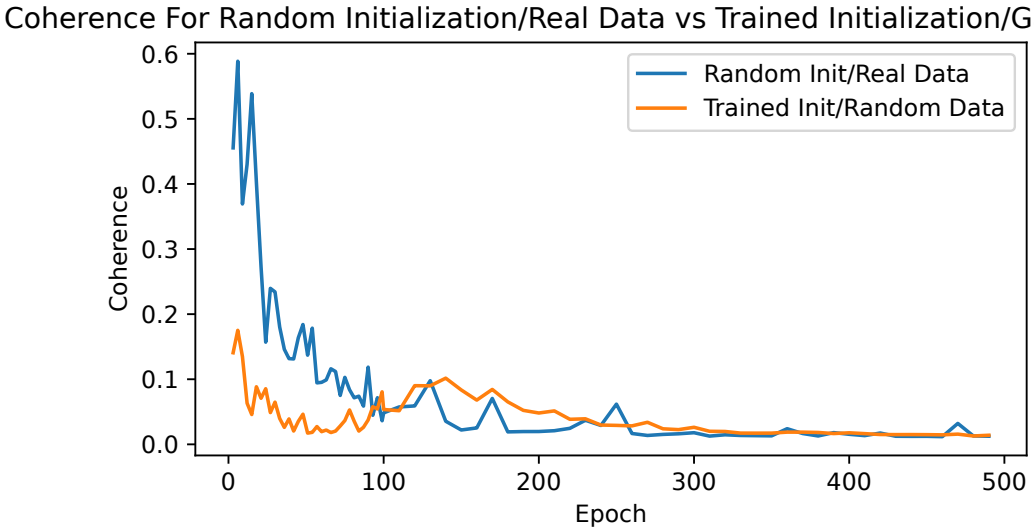

Figure 2.4: A comparison of the coherence measurements when training a randomly initialized model on real data vs starting with fully trained weights and training on random data. The goal is to test the hypothesis that coherence is only higher when training randomly initialized networks on real data as opposed to random data (as in Figure 2.1) because the optimizer for random data is closer in weight space to the random initialization point than the real data optimizer is. If this were the case, then a network previously trained on real data that is then trained on random data would experience comparatively high coherence during the training on random data. However, what we observe is that the coherence in this case is significantly lower than when a randomly initialized model is trained on real data, suggesting that high coherence is not merely due to how far the weights are moving in weight space.

the course of training.

3.1 Defining Diffs

Because coherence measures the expected value of the dot product between the gradients of two samples randomly chosen from the dataset, it is a network-level metric. Network pruning requires a scoring of each individual parameter in order to keep only the "best" parameters. To that end we introduce the "diff," a parameter-level metric (a metric with a different value for each parameter in a network) inspired by gradient coherence.

A first pass at adapting coherence to the parameter level is to simply sum up the element-wise gradients across the training dataset. This can be seen as the expected value of the gradient of the network in a particular direction, multiplied by the size of the dataset:

$$\text{diff}_p = \sum_{z \in D} \nabla \ell(z)_p = (m) \frac{\sum_{z \in D} \nabla \ell(z)_p}{m} = (m) \mathbb{E}[\nabla \ell(z)_p]$$

where p is the individual parameter, m is the number of samples in the dataset, and ℓ is the loss function.

However, this method is susceptible to large outliers in the dataset that generate large gradients in a direction opposite the majority of the data points. The intuition behind coherence suggests that it measures the extent to which each training sample helps the network better classify other samples in the dataset. Higher coherence means that there is a higher level of consensus amongst the various sample gradients as to which direction the network parameters need to go, no particular training sample can dominate the coherence calculation. If an outlier were to significantly skew the sum of the gradients it would render the parameter ranking useless as a proxy for coherence.

To fix the outlier problem and better capture the notion of a consensus amongst the various sample gradients, we look instead at the sign of the gradient. By adding up the signs of the gradient we obtain the difference between the number of samples that want a weight's

value to go up versus the number of samples that want the value to go down. The result is a measurement of the expected value of the sign of the gradient in a particular direction. For a network f, element p, dataset $\mathcal{D}$ and loss function ℓ, the equation is

$$\text{diff}_p = \frac{\sum_{d \in \mathcal{D}} \text{sign}(\nabla \ell(d)_p)}{|D|}.$$

We call the magnitude of this number the "diff." If the diffs are doing what we intend for them to do then diffs of higher magnitude should represent parameters whose values enjoy strong consensus from all the sample gradients. If the training data is indeed a good sample of the target distribution we are trying to learn with our network, then for a high diff parameter (where most of the training samples agree on what the parameter's value should be) it is likely that the parameter's value would help the network correctly classify a sample randomly drawn from the target distribution. On the other hand, for a parameter whose diff is near zero, a random sample from the target distribution would be equally likely to want the parameter value to go up or go down, meaning that the parameter's value is less likely to help the network accurately predict the correct label. This reasoning leads to the conclusion that parameters whose diffs are large have values that help a network generalize better to the target distribution, and parameters with low diffs primarily inject noise into the network's labeling process.

With this in mind, diffs seem to be a terrific candidate for ranking the usefulness of network parameters, a key step towards effective network pruning (Molchanov et al., 2019). A network composed of predominantly high diff elements should exhibit superior generalization compared to a network composed of low diff elements. There even seems to be the possibility that accuracy would increase as a result of diff-based pruning because of the removal of parameters that don't generalize well, parameters whose values are overfit to outliers in the dataset. In this paper we test this hypothesis by comparing diff-based pruning (using diffs) to magnitude-based pruning and random pruning. But first we investigate some of the basic properties of diffs.

When we calculate diffs, we do so by freezing the model at the end of a single epoch of training and then calculating the gradients for the entire dataset without updating the weights between samples.

3.2 Diff Histograms

Figures 3.1 and 3.2 show the distribution of the diffs for various layers of the ResNet50 and ViT networks. There are two key observations to make. The first is that the diffs are roughly normally distributed and mean centered. This means that the majority of parameters have a diff near zero, meaning they were adjusted in both directions almost evenly. This means that whatever the parameter's value is, both halves of the dataset are in disagreement about how the parameter should move, as their gradients point in opposite directions. This implies that the parameter's value need not change much to improve classification.

However in the case of relatively few parameters at the edge of the distribution with high diffs, most of the samples in the dataset "agree" that the parameter's value should move in a given direction. After this parameter has been trained in that direction, when the network encounters another sample, it is likely that the parameter's value will help the network classify that next sample because the sample is statistically likely to be similar to the rest of the data in the dataset, most of which agreed on which direction the parameter should move. The fact that high diffs are limited to a minority of the parameters could potentially imply that a majority of the network's generalizable learning is done by a minority of its parameters, once again suggesting that diff-based pruning is a reasonable thing to try.

The second key observation is that the diffs have a larger standard deviation when calculated during an epoch with high coherence as opposed to low coherence. This link between high coherence and large diffs is evidence that diffs are a correct quantization of coherence.

A third observation, one which urges caution when comparing diffs to coherence, is that the diffs for the networks trained on Gaussian data do not necessarily have smaller diffs. If it were truly the case that generalization large diffs are both direct results of large coherence,

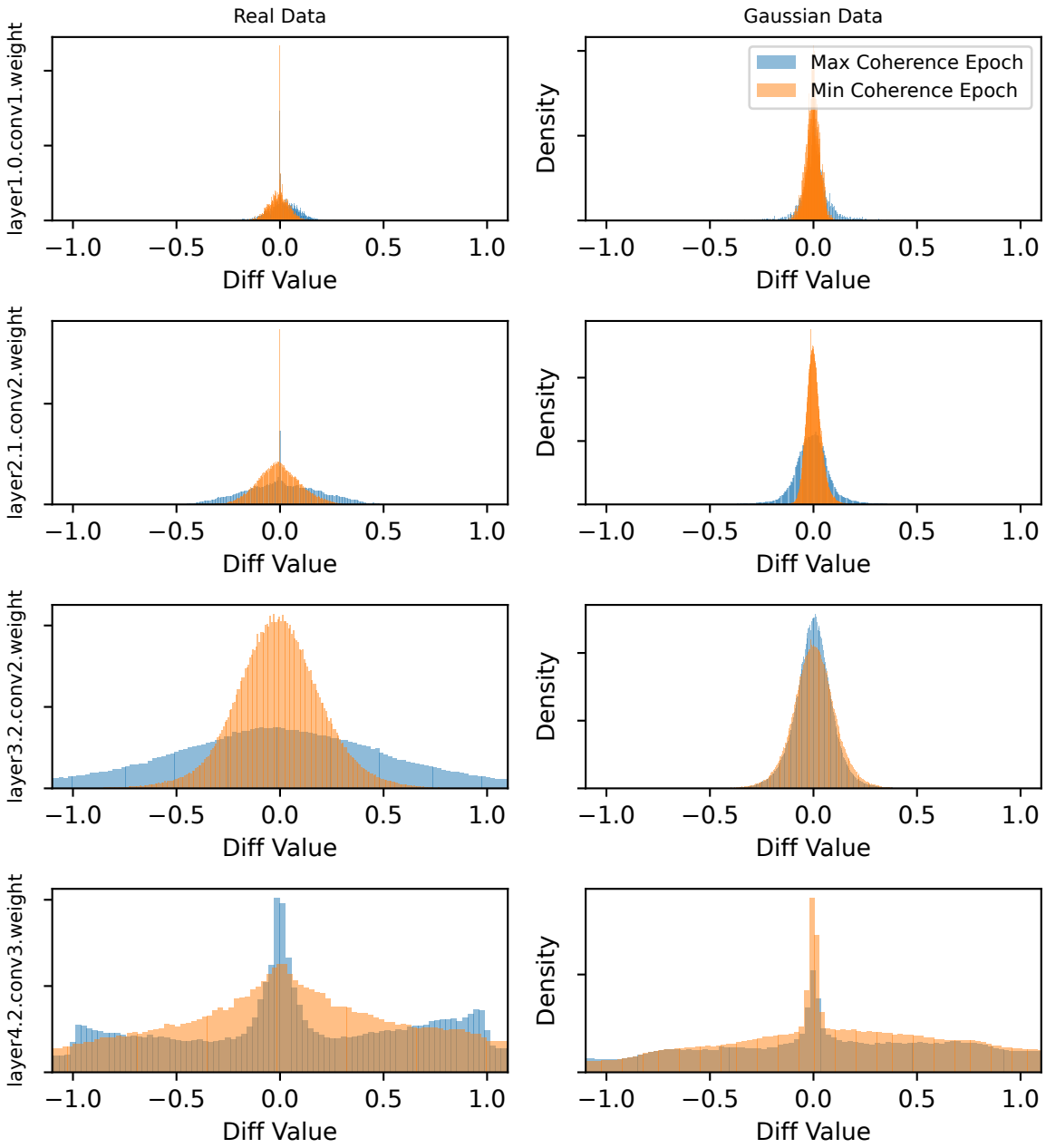

Figure 3.1: Histograms showing the distribution of diffs for various layers of ResNet50. The column on the left shows diffs for a network trained on real data, and the column on the right shows diffs for a network trained on Gaussian data. Each row gives the diffs for a different layer of the network. Diffs are calculated for both the epoch with the highest measured gradient coherence as well as the lowest gradient coherence. These plots show that diffs calculated during the epoch when coherence was highest are larger than diffs calculated when coherence was smallest. This correlation between average diff magnitudes and coherence measurement strengthens the case that diffs are a useful parameter-wise approximation of coherence.

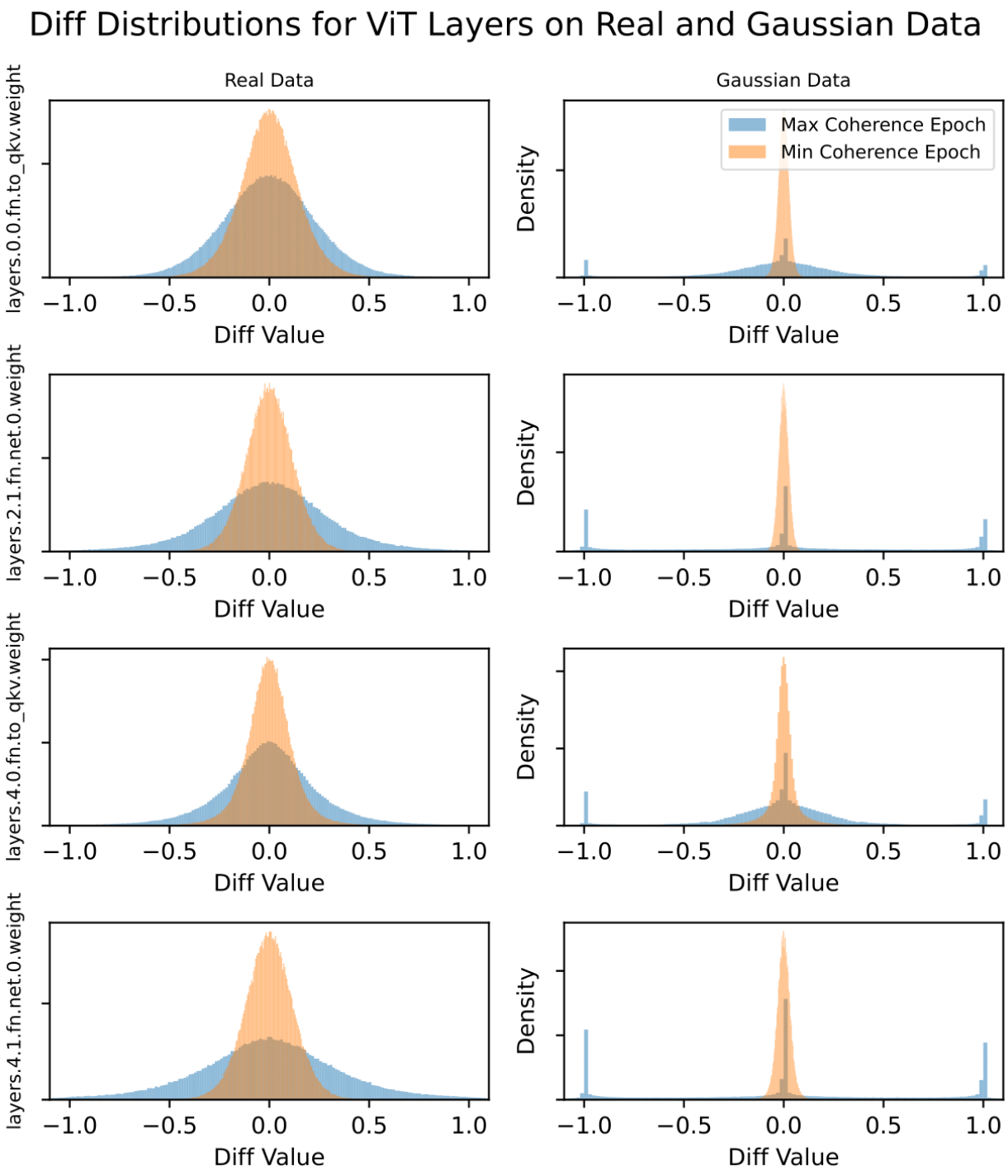

Figure 3.2: The same plots as 3.1 for ViT instead of ResNet50. Again, we see that average diff magnitude is larger when coherence is larger, suggesting that diff magnitudes are a good parameter-wise approximation of coherence.

we would expect that Gaussian data would generate very small diffs. To the contrary, memorizing data in a Gaussian network generates very large diffs. In particular, note the spikes at the edges of the Gaussian diff histograms for the ViT in Figure 3.2. Those spikes occur for the parameters where every single gradient pointed in the same direction.

We are then left with two contradictory pieces of evidence. Diffs are larger when coherence is higher, as is expected if diffs are tied to coherence. At the same time, diffs are not small when coherence is small during training on Gaussian data. It may simply be that the size and distribution of diffs is not itself an indicator of generalization, meaning that we can't look at a plot of diffs and tell whether the network will be able to generalize. This is because diff distributions are similar for both real and random training data. However, it is possible that in the case where we train on real data and generalization does occur, diff magnitudes tell us which parameters most contributed to the generalization. In hopes that this is the case, we investigated the usefulness of diffs for network pruning. But first, we will look at how the values of parameters with high and low diffs evolve over time.

3.3 High-Diff and Low-Diff Parameters During Training

One concern with the calculation of diffs is that even if it is not the case for coherence in general (as was shown in Section 2.2) perhaps high diff parameters are simply the ones that need to move further from their initialization point than low diff parameters. If this is true, then diff magnitude is correlated more strongly with weight magnitude than coherence, and diff-based pruning is just a convoluted form of magnitude pruning.

Figures 3.3 and 3.4 refute that hypothesis by showing no discernible qualitative difference between the evolution of parameter values with high and low diffs.

Now that we have shown that diffs are not a proxy for magnitude, we will examine the relationship between the width of hidden layers in shallow MLPs and the coherence of the resulting networks.

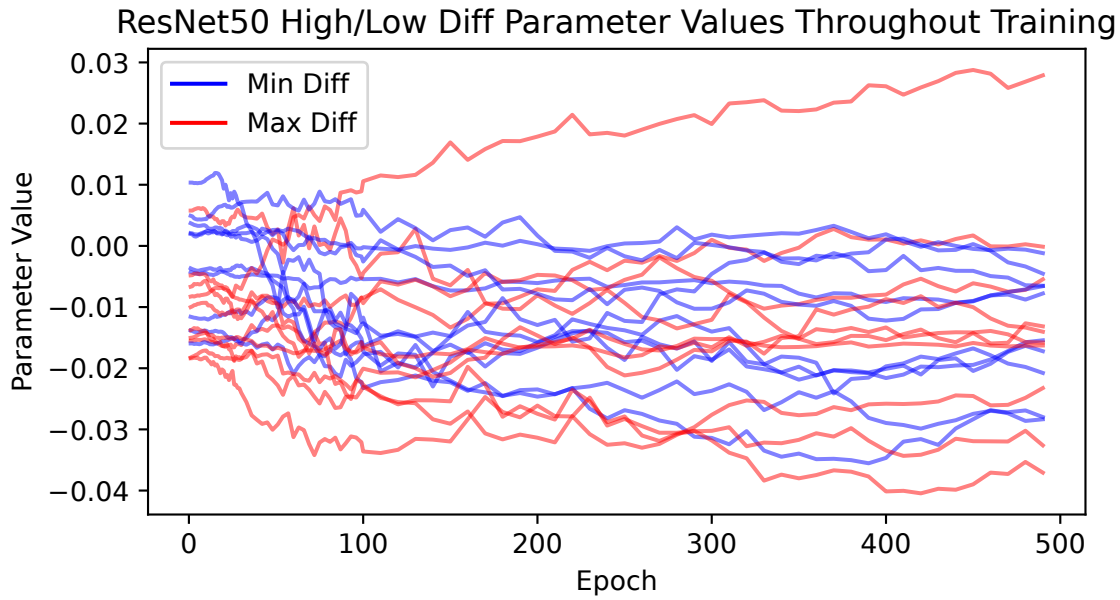

Figure 3.3: The evolution of parameter values for the top 10 high and low diff ResNet50 parameters throughout training on CIFAR-10 data. The high diff parameters do not seem to consistently migrate further than the low value parameters, meaning there must be another mechanism responsible for the diff discrepancy, such as a difference in coherence.

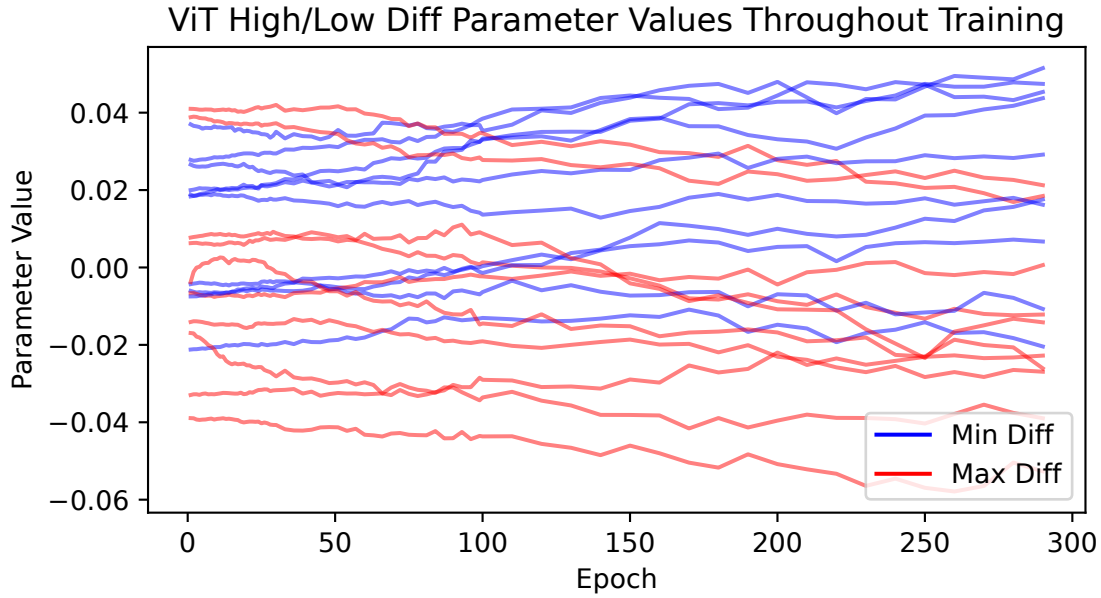

Figure 3.4: The same plot as 3.3 but for a ViT network instead of a ResNet50 network. Again, there is no discernible qualitative difference between the evolution of the two sets of parameters.

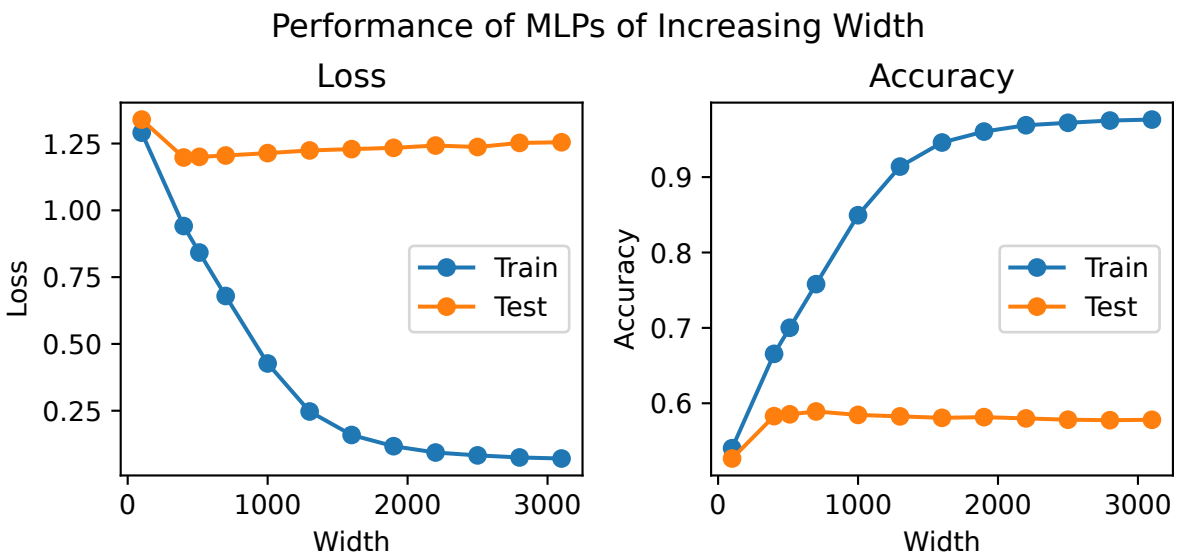

Figure 4.1: The train/test loss and accuracy of fully-connected networks of increasing size. As expected, the larger networks are able to perfectly fit the training data.

Chapter 4. The Influence of Width on Coherence and Generalization in Linear Networks

In this chapter we will examine fully-connected networks with 3 hidden layers of varying width trained on CIFAR-10 data. We will study the affect of width on the network's performance, generalization gap, and coherence. We will also examine the diffs associated with fully-connected networks of various widths, and what affect network width has on them. The purpose of this chapter is to establish a baseline of fully-connected network characteristics as width increases so that we can compare the results with those of pruned networks.

4.1 Performance vs Width

The first question we ask is how networks of various sizes perform at classifying CIFAR-10 images. Neyshabur et al. (2015) showed that increasing the size of hidden layers in a fully-connected network strictly improved performance. Figure 4.1 replicates this finding, showing

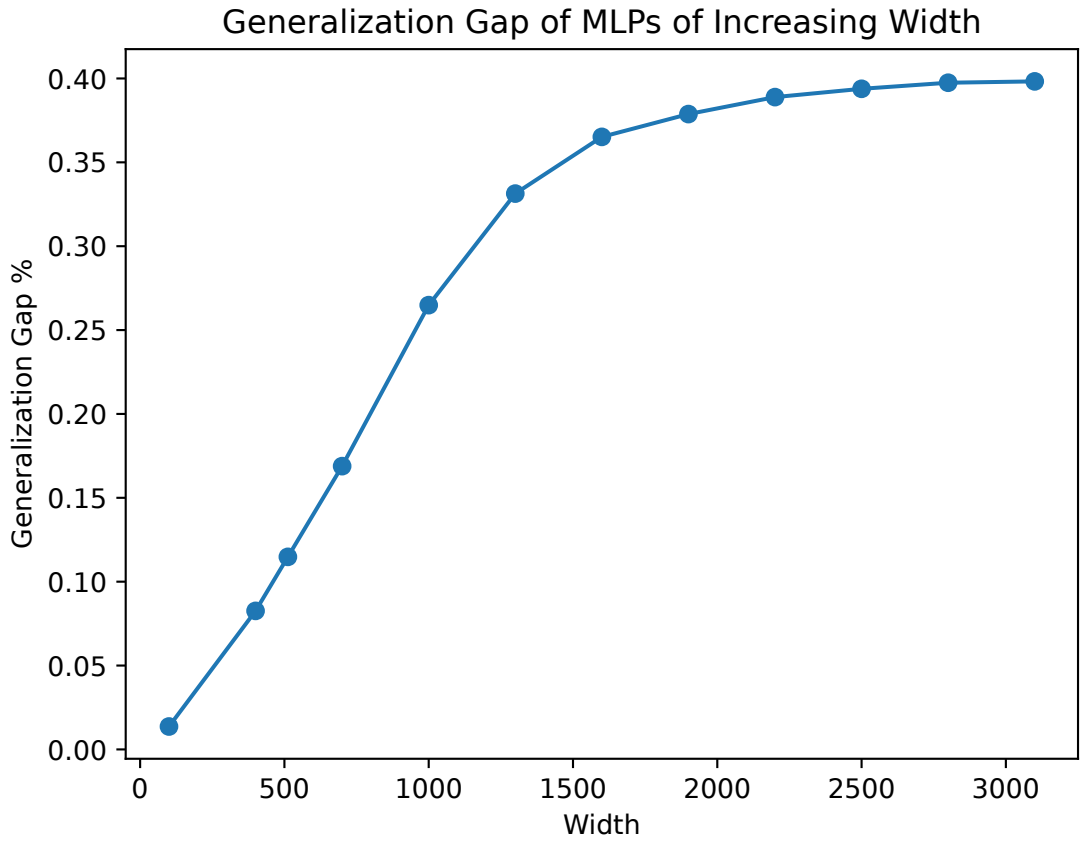

Figure 4.2: The generalization gap (training accuracy − test accuracy) of fully-connected networks as a function of the hidden layer width. In keeping with Neyshabur et al. (2015), the size of the generalization gap does not grow indefinitely with increasing network size, but instead levels off.

that as the number of parameters increases the training loss goes to zero and the training accuracy nears 100%, while the test loss does not increase with the increasing overfitting as size increases.

Figure 4.2 plots the generalization gap as a function of network width. It climbs steadily until 1500 neurons per hidden layer, at which point it levels out because perfect train accuracy is reached and test accuracy has leveled out. This leveling out of the generalization gap is line with Neyshabur et al. (2015).

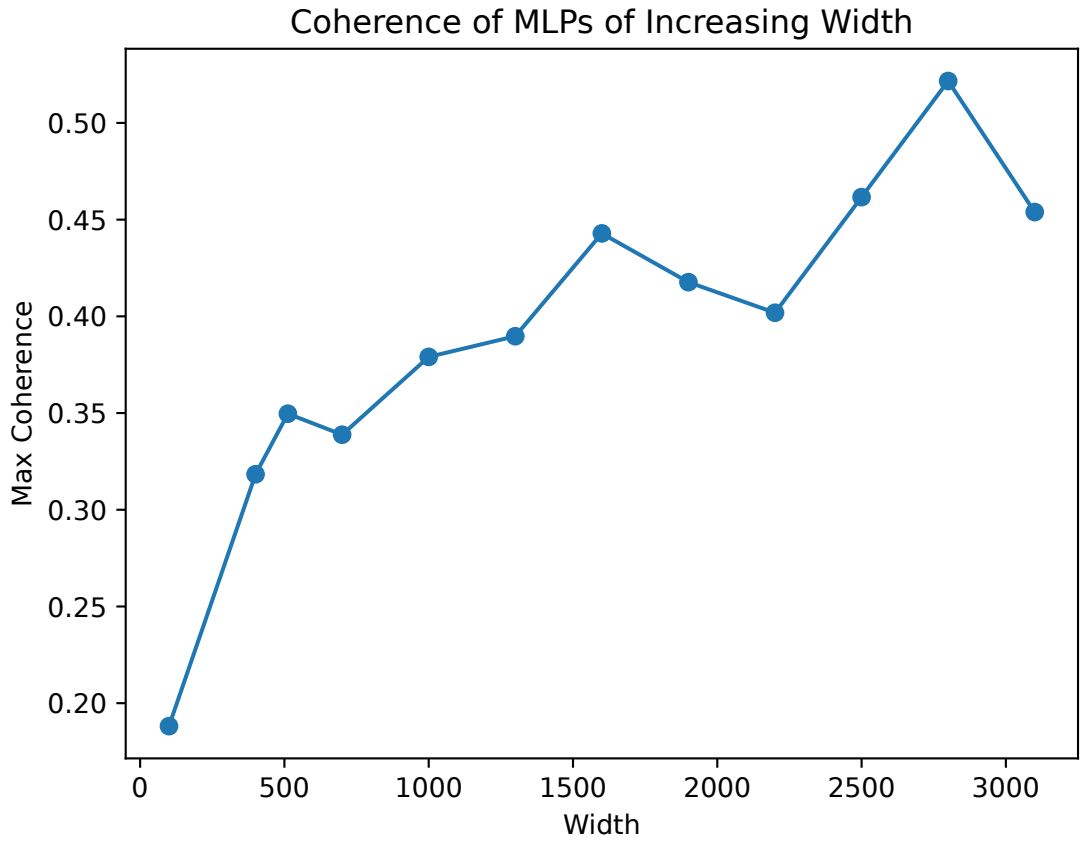

Figure 4.3: Max coherence of fully-connected networks plotted as a function of the width of the hidden layers. This plot suggests that a network's measured coherence is roughly proportional to its size, or at least that wider fully-connected networks demonstrate greater coherence than narrower ones.

4.2 Coherence and Diffs vs Width

Having seen the increasing performance and widening generalization gap, we ask how the width of the networks affects coherence. Figure 4.3 plots the largest coherence measurements for each network width.

An initial observation to make is that the coherence is much smaller for these fully-connected networks than it was for the ResNet50 and ViT networks trained on real data. Coherence seems to increase as network width increases. One hypothesis that would explain why coherence increases with size is that the larger networks are able to search a larger space of parameter value paths and find those that generalize better. These parameters generalize so well that they have very strong gradients pointing in the same direction across all the training data, pulling up the average coherence with them. There is, however, contradictory evidence to consider. According to the coherent gradients hypothesis, if coherence is rising then the network should generalize better. However, the generalization gap widens with increased width instead of shrinking. It is possible that the variation in size between the networks is responsible for both increasing the size of the generalization gap and increasing coherence, meaning their simultaneous increases are a correlative rather than causative. This would suggest that (at least in fully-connected networks) comparing coherence is best done between networks of similar size, because variations in coherence due to generalization are drowned out by variations caused by differences in size.

Finally, we examine the effect of increasing network width on the network's diffs. Figure 4.4 shows that the diffs do seem to decrease in standard deviation as width increases. Strangely, the diff distributions are at their narrowest with width 2800, which is when coherence is at its highest. A potential explanation for this is that with larger networks, more of the "work" of classification can be done in the earlier layers and so there is less meaningful information being captured in the values of the parameters in the later layers.

Now that we have established a baseline for the performance, coherence, and diffs of fully-connected networks as a function of hidden layer width, we are prepared to study the

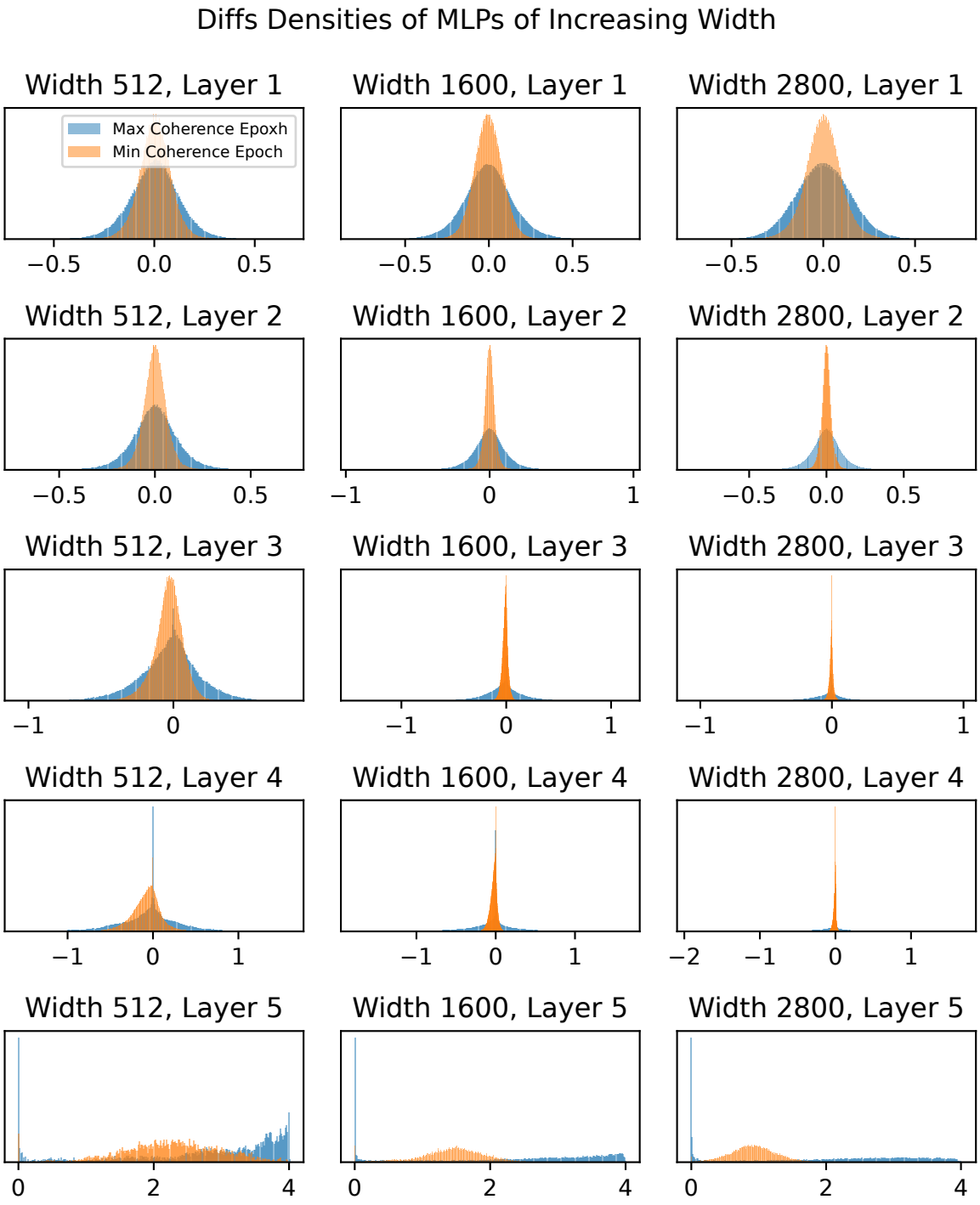

Figure 4.4: Diff distributions for various fully-connected network widths, calculated once when coherence was at its maximum, and again when coherence was at its minimum. Each column contains diffs for a different width, and each row contains diffs for a different layer. Once again, average diff magnitudes are larger when coherence is larger, and it is smaller when coherence is small. The distribution of the diffs seems to narrow as the width of the network increases. In contrast, coherence tends to rise as network size increases. This casts some doubt on the connection between diffs and coherence.

effects of pruning on these increasingly wide fully-connected networks.

Chapter 5. Pruning Linear Networks

We have now arrived at the point we have been building up to; we will now try to prune neural networks using diffs. We will begin by pruning the fully-connected networks similar to the ones in the previous section. We will combine this analysis of diff-based pruning with a comparison to traditionally pruned networks and dense networks of the same size. We will conclude with an examination of the effect of the size of the parent network on a sparse network's accuracy; while holding the number of non-zero parameters constant, does increasing the size of the parent network improve performance? How much?

5.1 Literature Review on Network Pruning

State of the art neural networks keep getting bigger and bigger, becoming ever more compute hungry. At the same time, these networks are becoming significantly more useful which incentivizes their use on lower-powered devices at the edge, for example smartphones. Much research has been done on methods for capturing all of the power of a large model on a smaller model. While distillation and fine-tuning have had remarkable success, the holy grail technique for shrinking a neural network is pruning. Pruning means replacing a larger network with a sparse subnetwork. A sparse subnetwork is the network that remains when a significant fraction of the original network's weights are clamped at zero. These sparse networks are less compute intensive at both training and inference time. Ideally these sparse networks would retain most if not all of the parent network's performances. Unfortunately, a review of the literature suggests that we have yet to discover a pruning method that works on complex datasets and can produce sparse networks that compete with their dense counterparts.

We know that sparse, performant subnetworks exist. Frankle and Carbin (2019) is a

seminal work in the field that introduces the "Lottery Ticket Hypothesis." They state the Lottery Ticket Hypothesis as: "A randomly-initialized, dense neural network contains a subnetwork that is initialized such that—when trained in isolation—it can match the test accuracy of the original network after training for at most the same number of iterations." They then develop a technique for discovering these "winning tickets," or performant sparse subnetworks, using a technique called Iterative Magnitude Pruning (IMP). IMP prunes the network by starting with the full network and training the parameters in the network and then removing a percentage of the weights, repeating the process until a desired level of sparsity is reached. The weights to be removed are the ones with the smallest magnitudes. Interestingly, the subnetworks obtained this way can train to higher accuracies in fewer epochs than the original networks. This suggests that many of the parameters in the original networks are at best useless, and potentially even harmful to the functioning of the network.

One final feature of the networks identified by IMP is that they are dependent on starting with the same initialization as the original network. That is, if the network is randomly re-initialized then the sparse subnetwork will no longer perform as well. This suggests that part of the magic of IMP is identifying a subnetwork that has been initialized in a fortuitous region of weight-space, perhaps one near a particularly good minimum.

Frankle et al. (2020) improves on the original IMP algorithm by searching for networks that could have been obtained by pruning early in training (0.1% to 7% through) instead of at initialization. Instead of resetting the weights of unpruned parameters back to what they were at initialization, they are rewound and assigned their values from some point early in the training process. This modification allows IMP to discover subnetworks for deeper networks that it was previously unable to prune.

Although these initial results are promising, they might not be transferable out of the realm of simple CNNs on MNIST and CIFAR-10. Gale et al. (2019) found that when IMP and similar techniques are scaled and used to train Transformers and ResNet50 on a variety of tasks the sparse networks cannot be trained to a performance level on par with the original

network. They actually found that simple magnitude pruning more performant when pruning these large networks, even if the pruned networks do not achieve the same performance as the originals.

Molchanov et al. (2019) explores an interesting implication of the existence of sparse trainable networks, attempting to estimate the importance of each neuron to the network's capacity. They define importance as "the squared change in loss induced by removing a specific filter (neuron) from the network." A Taylor expansion is applied to the square change in loss to get the parameter-wise importance estimation. They then iteratively remove the least important parameters. While they were not able to achieve parity with unpruned models for large networks, they were able to improve upon other methods and get similar error with fewer parameters and fewer FLOPS of computation.

Another issue with IMP is that while it is successful, it is very expensive, requiring a significant amount a re-training. Much of the literature following the Lottery Ticket paper seeks to find pruning mechanisms that require significantly less computation. Once the network is fully trained, magnitude pruning seems to remain the most effective method. These new methods seek to achieve parity with dense networks while training "from sparsity," meaning that the network begins training as a sparse network and stays sparse throughout training. Some approaches modify the architecture while training occurs to redistribute the sparse parameters to sections of the network where they might be more useful. The following two algorithms follow this pattern.

Evci et al. (2021) and their algorithm RigL is one such attempt at training from sparsity. They start with a random sparse network and then at regular intervals update it by dropping low magnitude weights and re-growing new weights. The trick is that the weights are re-grown by selecting the neuron connections where the gradient is the highest. The idea is that large gradients mean that a parameter's value contains important information. They achieve performance superior to previous pruning methods that trained with fix sparsity from scratch.

Dettmers and Zettlemoyer (2019) demonstrates "the possibility of training sparse networks that rival the performance of their dense counterparts with a single training run - no re-training is required." Their algorithm, sparse momentum, uses exponentially smoothed gradients to perform parameter importance estimation. This is similar to Molchanov et al. (2019) where they used a Taylor expansion of the loss function (whose first order approximation is just the gradient) estimate importance. The difference in their approach is that they focus on mean momentum (gradient average) per layer, and then follow an approach similar to Evci et al. (2021) where the prune and re-grow neurons. They prune 50% from each layer (using magnitude as a criteria) and then use the mean momentum measurements to apportion the new neurons. While they are still not as accurate as dense networks, they do perform well.

We will also mention several more attempts at pruning at initialization. The difference with these is that they don't modify the network during training like RigL and sparse momentum do. They instead seek to do all the pruning before training even begins.

Lee et al. (2019) introduces and algorithm they call SNIP (Single-Shot Network Pruning). As with other pruning algorithms, they attempt to calculate a "saliency measure" to guide them in pruning the network. They also use the gradients to derive this measure. They measure sensitivity as the largest normalized gradients at initialization. Thus they pruned the parameters with the smallest normalized gradient magnitudes. One important feature of this method is that it does take the dataset into consideration, so the subnetwork that it identifies is specifically tuned to the problem domain.

Where the scoring function used by SNIP is a first-order method based on the magnitude of the gradient of the loss function $\mathcal{L}$ with respect to a particular parameter θ.

$$S(\theta) = |\frac{\partial \mathcal{L}}{\partial \theta}|.$$

The goal of this method is to score parameters based the magnitude of their respective gradients. Parameters with larger gradients are ones where the loss is more "sensitive" to that parameter being removed. Pruning only parameters with low sensitivity scores is

intended to preserve as much of the network's performance as possible.

Wang et al. (2020) introduces a new algorithm called GraSP which computes scores as

$$S(\theta) = -\theta \cdot H\frac{\partial \mathcal{L}}{\partial \theta}$$

where H is the Hessian of the loss function with respect to θ.

They do this because they want to preserve "gradient flow," the magnitude of the gradients throughout training. They believe that pruning with SNIP ignores how the parameters interact with each other to produce gradients, and as a result the gradients are too muted during training, hindering optimization. Pruning based on the Hessian is intended to capture the interaction between the neurons and improve the total amount of gradient during training. GraSP achieved similar results to SNIP, besting it on some network/pruning ratio combinations and being surpassed by it for others.

Tanaka et al. (2020) examines whether or not it is possible to prune a network without even looking at the data. Their algorithm, SynFlow, seeks to avoid what they call "layer collapse," in which a single layer gets pruned too much and inhibits the network's training. They show that SNIP and GraSP are more prone to this kind issue. In addition to their empirical evidence, they show theoretically that any score based on gradient magnitudes will be subject to the same issue. This is why their pruning method is data-independent. They instead use Synaptic Flow, which is a generalization of scoring by magnitude. On CIFAR-10, CIFAR-100, and Tiny ImageNet, SynFlow was able to out-perform GraSP and SNIP.

Frankle et al. (2021) makes the point that despite the fact that these three measures (SNIP, GraSP, and SynFlow) perform better than random pruning, they are still not as good as magnitude pruning after training, which is again surpassed by IMP. The authors demonstrate that if you shuffle which weights are pruned within each layer yields just as good of results. This suggests that the pruning methods are only effective at apportioning layer-wise pruning percentages, and don't actually identify which weights within the layers are most important. Interestingly, this means that SynFlow's lack of use of the dataset when pruning doesn't harm it. It is unclear if that means that data is unnecessary or (more likely)

that there has yet to be an algorithm that makes full use of the information gained from the dataset.

Further muddying the waters, Blalock et al. (2020) surveyed 81 papers on pruning and found that a lack of consistent benchmarks makes it difficult to compare different pruning methods. It is hard to see how much progress has been made in the field of pruning.

Alizadeh et al. (2022) aims to improve on prior pruning methods by taking into account the trainability of the network identified by a given saliency criteria. They do this by incorporating higher order gradients, or meta-gradients. The estimate the effect of pruning a neuron by performing multiple iterations of backpropagation and weight updates and then backpropagating the change in the loss function through the entire computation graph. This change favors parameters that have a large impact on the loss function after several steps of training, even if they might not have demonstrated large gradients at initialization. This method outperformed other methods of pruning at initialization.

Sreenivasan et al. (2022) is a more recent attempt to do better at pruning at initialization. Instead of relying on gradient based methods, they search for a sparse subnetwork that has a non-trivial degree of accuracy pre-training. They then fine-tune that subnetwork to more closely match the accuracy of the final network.

To summarize, the current state of the field is that there are no methods of pruning that reliably compete with dense networks outside of toy architectures and datasets. Iterative pruning is strong in toy situations but expensive and doesn't scale well. There are many attempts at pruning at initialization, many of them based on gradient magnitudes. However, none of them are significantly better than magnitude pruning. Pruning, and especially pruning at or near initialization, remains an open problem.

5.2 Why Coherence Based Pruning?

One of the most fundamental challenges in network pruning is parameter importance estimation. Many of the experiments in the preceding literature review were based on trying

different methods for deciding which parameters are more important than others. To this, we add coherence-inspired diff pruning. As we have alluded to before, the logic behind diff-based pruning is that if high coherence does in fact indicate generalization, and large diffs are an indicator of a parameter contributing to high coherence, then pruning a network to contain only those parameters with large diffs should reveal an effective subnetwork. This focus on consensus rather than magnitude makes this pruning method robust to outliers in the dataset. Nelson (2022) found that coherence doesn't seem to correlate with final low training loss. However, diffs might help us understand which parameters most contribute to low loss, allowing us to create performant sparse networks.

Another nice property of diff pruning is that the information needed to rank parameters can be obtained much earlier in the training process. Where magnitude pruning requires fully training the model to have access to parameter magnitudes, peak coherence is typically achieved relatively early in the training process when training on real data. This means that the training process can be run for fewer epochs before a mask can be obtained.

Compared to methods that seek to prune at initialization such as SynFlow, diff-based pruning prunes based on the specific dataset that is to be trained on. When doing deep learning, the final model is always a function of the architecture, optimization algorithm, and the dataset. It seems like a pruning method that uses information about the dataset should outperform one that doesn't. Diff based pruning takes advantage of this information without too much of the extra training burden that pruning at initialization methods seek to avoid.

5.3 Pruning Increasingly Wide Linear Networks To 10% Sparsity

In our first pruning experiment we start with the same set of widths we used in our investigation of coherence as a function of width. Due to the quadratic way that parameter count increases with width, for each width we compute a second width that results in a network

with 10x the parameter count. For input vectors of length 3072 (like CIFAR-10 images) and 10 output classes, scaling a network of width w by a factor r to get the scaled width w' is calculated with the equation:

$$w' = w + (\frac{1}{3})(\sqrt{3rw(3w + 3082) + 2374681} - 3w - 1541).$$

We then use one of three masking techniques to prune the network to 10% sparsity, recovering the original number of parameters in the dense network up to a difference in the number of bias parameters, which are not pruned. These pruned networks are trained for 200 epochs with Adam using a learning rate of 1e$-$4. Importantly, we do not prune every single layer of the network. We skip layers that are small, so as not to create a bottleneck in the architecture. For ResNet50 this means pruning only the convolutional layers and the larger shortcut layers. We also skip pruning batch norm and bias layers.

The first masking technique that we use is diff-based masking, where we select only the parameters with the largest diff magnitudes. The second is magnitude based masking, where we select the parameters with the largest magnitudes. The third mask is random masking, where the decision of which weights to keep is completely random.

To select the diff mask we calculate the diffs using the epoch for which the highest coherence was measured during training. While we cannot know which epoch exhibited the highest coherence until the network is fully trained, the tendency of coherence to peak early when training on real data means that we should be able to identify the max coherence epoch relatively early in the training process, saving compute. For the magnitude mask, we use the network weights at the end of training.

Each network is trained from a random initialization. We do not keep with the Frankle and Carbin (2019)'s procedure of re-using the same exact initialization between the original training run and the masked training run. Our aim is to discover whether or not pruning can identify a subnetwork whose innate structure grants it performance, rather than finding a particular initialization that works well. Our pruning methods are intended to be a search in architectural design space that may help to inform future architectural design.

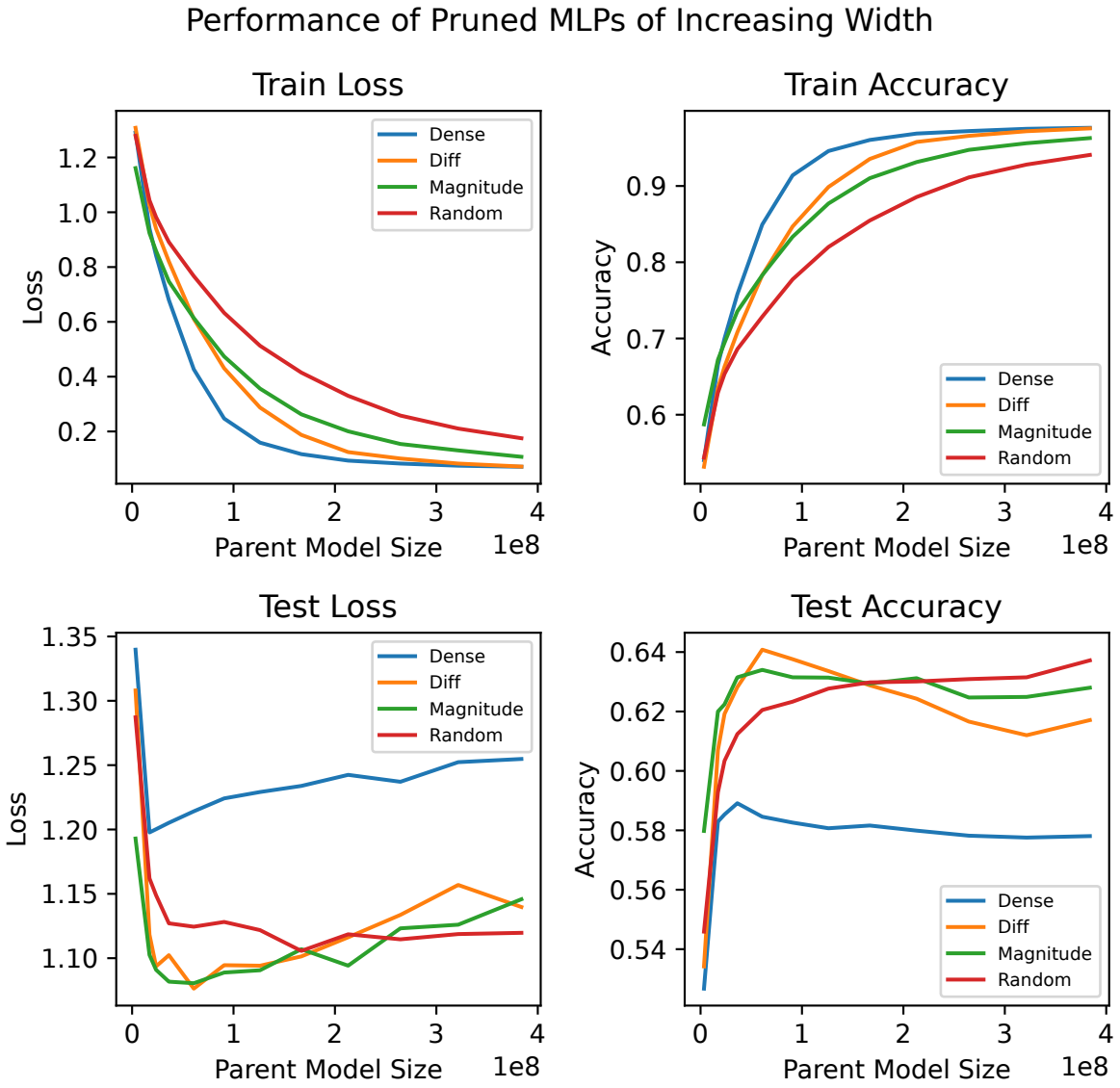

Figure 5.1: Performance of pruned fully-connected networks. Each parent network was pruned to 10% sparsity using one of several difference pruning techniques and then re-initialized and trained from scratch. The line labeled "none" represents a dense network with the same number of parameters as the sparse networks. It is clearly demonstrated that the sparse networks derived from larger parent networks are more performant than the dense network of comparable size.

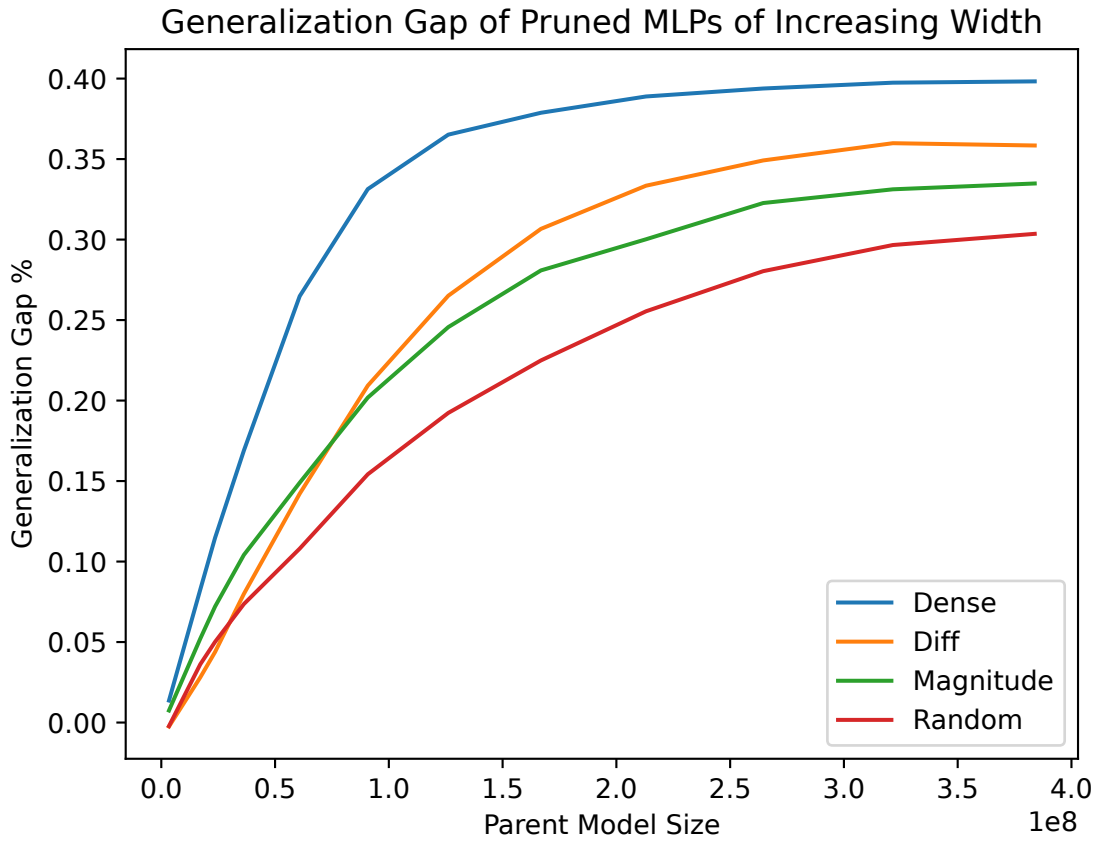

Figure 5.2: Generalization gap for pruned fully-connected networks.Each parent network was pruned to 10% sparsity using one of several difference pruning techniques and then re-initialized and trained from scratch. The line labeled "none" represents a dense network with the same number of parameters as the sparse networks. All sparse networks generalized better than the dense network, but diff-based pruning generalized significantly better than magnitude based pruning when the network size was small. Interesting, random pruning generalized the best.

Figure 5.1 shows the loss and accuracy on train and test sets for each type of pruning, as well as for the original dense network. We can clearly see that the dense network does the most overfitting - it has the best performance on the training data and the worst performance on the test data. Surprisingly, the randomly pruned network performed very well as network size increased, achieving the highest accuracy at large widths. The coherence and magnitude pruning methods required significantly smaller parent networks to reach the highest levels of accuracy. Magnitude pruning seems to out-perform diff-based pruning in this setting.

Figure 5.2 analyzes the performance of these pruned models through the lens of the

generalization gap. Here we see that, especially in smaller networks, diff-based pruning does actually outperform magnitude based pruning. This suggests that our coherence approach does in fact bias networks towards better generalization. This is a promising result. It remains surprising that random pruning is so effective at controlling the generalization gap, even more effective than diff-based pruning. Further research is needed to understand this.

5.4 Fixed Size Networks With Increasingly Large Parent Networks

In the previous section we help the level of sparsity constant at 10%, meaning that as the parent networks grew in size so did the sparse child networks. Here we investigate sparse networks from a slightly different perspective - if we hold the number of non-zero parameters in a sparse child network constant, how does its performance change as we increase the size of the parent network it was derived from? In other words, is it better to prune a comparatively smaller parent network to a moderate level of sparsity, or to prune a much larger parent network to a much higher level of sparsity?

We started with a fully-connected network with 3 hidden layers, as in the previous experiments. The base network had a hidden layer width of 512 neurons, totalling 2.3M parameters. We then trained a series of increasingly large parent networks, up to 62x larger than the original network. Each of these larger networks was pruned so that the resulting sparse network had the same number of nonzero parameters as the original 512 width dense network. The training protocol was the same, re-initialization followed by training for 200 epochs with Adam and a learning rate of 1e$-$4.

Figure 5.3 gives the results of this experiment. Because the number of nonzero parameters is equivalent to a dense network with width 512, we should expect coherence and magnitude pruning to dominate like the did in the low width setting in the previous experiment. The results indicate that this is true. In addition, we see that there is in fact an increase in accuracy with the scale of the parent model, and the top-end accuracy achieved was similar

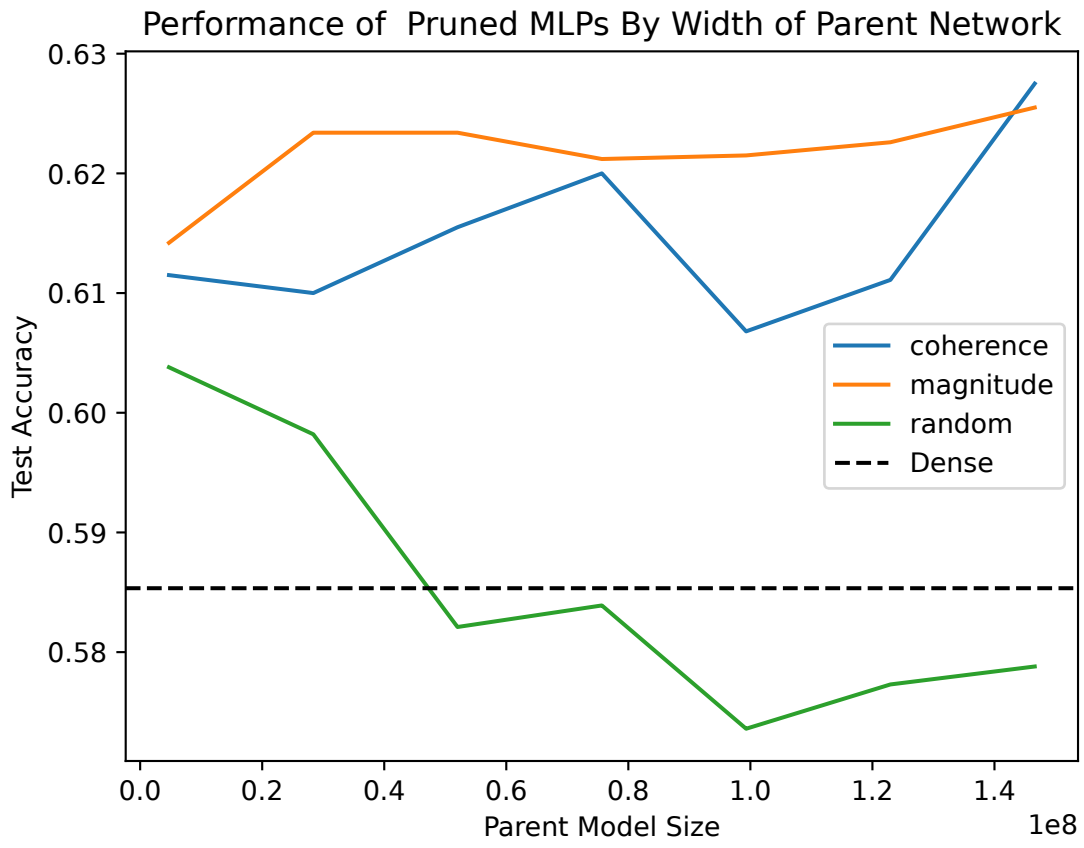

Figure 5.3: Performance of sparse networks derived from increasingly large parent networks. While the number of parameters in the dense parent networks increased, each was pruned to a different sparsity level so that each parent network yielded a sparse child network of equivalent nonzero parameter count. The figure shows that the performance of the sparse network can be improved simply be increasing the number of parameters in the parent network it is derived from. Once again, magnitude pruning seems to be outperforming diff-based pruning. In contrast to the previous experiment where the pruning level was kept constant at 10%, we can see that random pruning starts to struggle when the network is pruned too heavily.

to what was achieved with the 10% sparsity pruning of smaller models. The difference is that where the 10% sparsity model accuracy peaked when the parent network had 50M parameters, in this setting we needed a parent model size of between 60M and 100M parameters to achieve similar results, depending on the network size.

The conclusions that can be drawn from this chapter are quite striking, given that they are only shown to hold in this limited setting of fully-connected networks with 3 hidden layers. First, we found that sparse networks perform better than their counterparts of equivalent parameter count. Second, we find that scaling the parameter count of the parent network without increasing the number of nonzero parameters in the resulting sparse network will still improve accuracy. This means that efforts to prune a model should really be thought of as lying on two different axes. The first axis, which is traditionally the one investigated, is that of varying sparsity ratio. The second axis is the size of the parent network. Future attempts at creating sparse models should consider experimentation along both these axes to produce the most performant final model with the fewest parameters.

Chapter 6. Pruning Large Networks

Having demonstrated the effectiveness of pruning on simple fully-connected networks, we now turn our attention to the pruning of industrial-strength networks with more complex architectures. Here we will evaluate coherence, magnitude, and random pruning on the ViT and ResNet50 networks that we examined in the first two chapters.

6.1 Global Pruning

For our first experiments we used a global pruning strategy, in which different layers can be pruned to different sparsities based on the rankings of their respective parameters. This is the approach favored by most of the existing pruning literature, including Frankle and Carbin (2019). Because some pruning methods prune single layers too heavily, thus creating

a bottleneck, we establish a 2% sparsity floor for each individual layer, not pruning it further. We prune each network to 10% sparsity and then train it from initialization. As with our fully-connected networks, we do not preserve initializations between training runs. ResNet is trained with SGD and learning rate 0.5, while ViT uses Adam and learning rate 1e−4. Both are trained for 300 epochs.

In addition to employing masks which keep only the parameters with the highest coherence or magnitude, we test the efficacy of these pruning methods by employing masks that invert those parameter ranking. That is, if the "coherence_best" mask takes the parameters with the top 10% of diff magnitudes, the "coherence_worst" mask will keep only the parameters with the lowest diff parameter. This is also true for "magnitude_best" and "magnitude_worst." The "random_10" mask means that a random subset of 10% of parameters are kept.

Figure 6.1 list the results of this pruning. For the ResNet we see that magnitude pruning is most effective, and it is surprisingly followed by the "coherence_worst" mask. It seems that diff-based pruning actually has an inverse effect on the efficacy of training pruned ResNet models, and taking the parameters with small diffs is actually belonging to two different classes.

Figure 6.2 shows the results for the ViT, and surprisingly we see diff-based pruning as the clear winner. In fact, it matches the performance of the full dense network. This plot shows that there is a very clear distinction in how different networks respond to different pruning methods. We also see that random pruning actually outperforms magnitude pruning in this scenario.

6.2 Pruning By Layers

What explains the poor performance of diff-based pruning on ResNet? Figure 6.3 shows that coherence_worst pruning (as well as magnitude_best pruning) performed poorly because many of the layers were pruned all the way to the 2% sparsity minimum. Figure 6.4 shows

Performance of Pruned ResNet50

Masking Method	Train Accuracy	Test Accuracy
random_10	0.99816	0.9133
magnitude_best	0.997	0.9091
none	0.9988239999999999	0.89994
magnitude_worst	0.99728	0.8968
coherence_best	0.96358	0.1285
coherence_worst	0.1014	0.1

Figure 6.1: Results for pruning ResNet50 to 10% sparsity using various techniques. Layers were pruned non-uniformly, meaning layers with parameters that ranked higher in "importance" were pruned less than "unimportant" layers.

Performance of Pruned ViT

Masking Method	Train Accuracy	Test Accuracy
none	0.992684	0.7781399999999999
magnitude_best	0.98346	0.7752
coherence_worst	0.82108	0.7642
coherence_best	0.7723	0.7413
random_10	0.80194	0.7402
magnitude_worst	0.29234	0.2806

Figure 6.2: Results for pruning ViT to 10% sparsity using various techniques. Layers were pruned non-uniformly, meaning layers with parameters that ranked higher in "importance" were pruned less than "unimportant" layers.

that ViT, on the other hand, had its layers pruned much more evenly, without any layers hitting the sparsity minimum. It is unclear why coherence/diffs seem to be concentrated near the terminal layers of the network for ResNet but not for ViT.

Because disproportionate layer pruning seems to be responsible for the poor performance of diff-based pruning on ResNet, we re-run our pruning experiment with the modification that each layer is uniformly pruned to a 10% sparsity. Because parameter scores are only compared with other parameters in the same layer, we refer to this as "local" pruning.

Figure 6.5 shows that under uniform pruning, diff-based pruning results are no longer inverted, with the "worst" pruning prevailing over the "best" pruning. Another interesting takeaway from this table, however, is that 10% random pruning nearly matched global magnitude pruning for ResNet, which was the most successful masking strategy for ResNet. This result drives home just how effective random pruning can be, and should perhaps cause us to question our current understanding of pruning methods.

Figure 6.6 shows that diff-based pruning is still the best way to prune ViT, although with uniform layer pruning it was unable to achieve parity with the original dense network.

Chapter 7. Analysis and Conclusion

In this thesis we have covered a lot of ground and performed a lot experiments. We tested the coherent gradients hypothesis, introduced diffs, evaluated coherence on fully-connected networks, pruned fully-connected networks, and then pruned larger networks. In the Analysis section we will review and analyze the results of these experiments, highlighting the conclusions that can be drawn. In the Future Work and Conclusions section we will list the questions that remain unanswered.

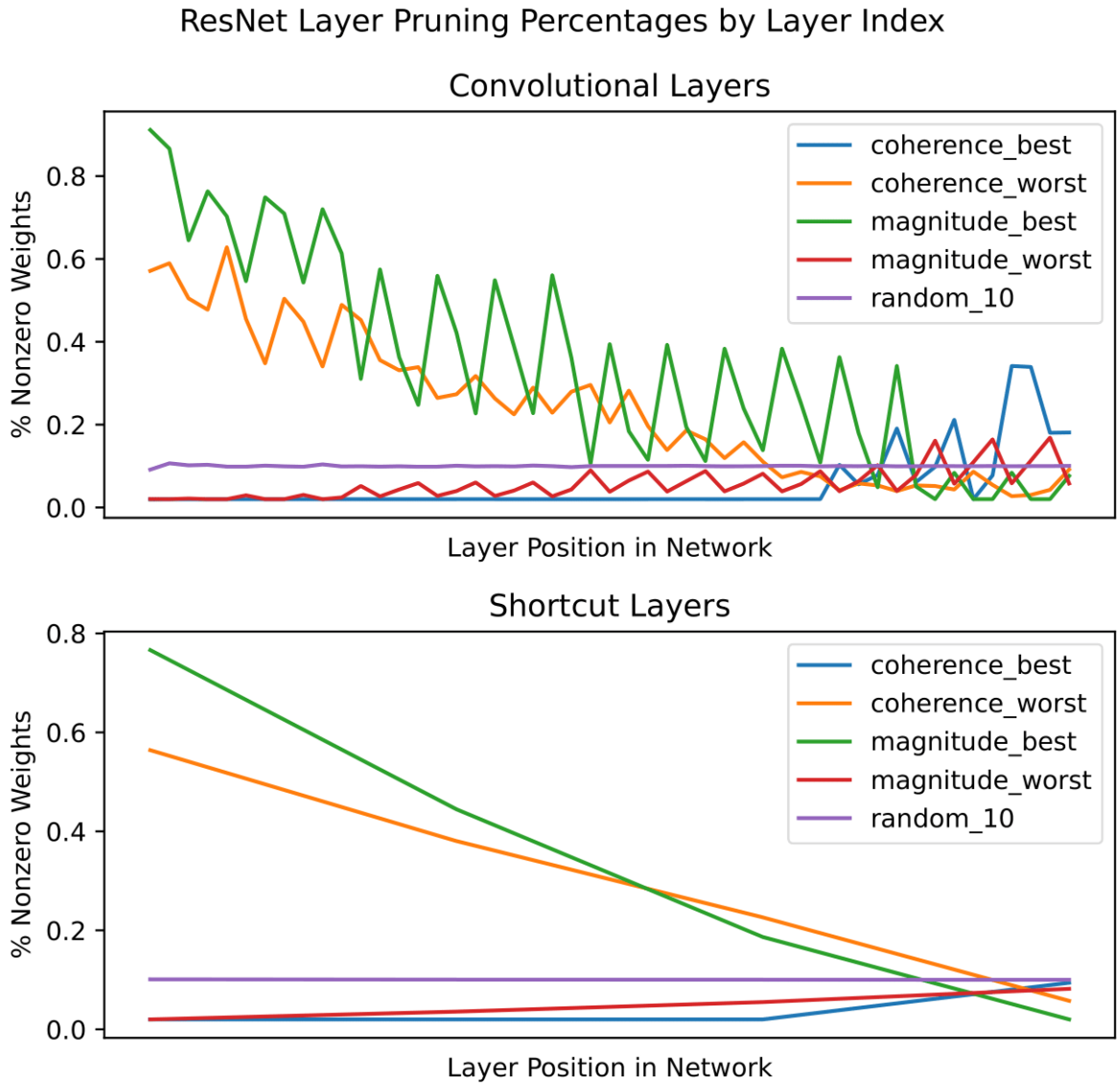

Figure 6.3: Layer-wise sparsity levels resulting from pruning ResNet50 with a variety of methods. ResNet50 layers are divided into two different groups, convolutional layers and shortcut layers. The x-axis represents increasing depth within the network. "coherence_best" and "magnitude_worst" severely pruned the early layers, which is likely responsible for the lesser performance of the resulting networks.

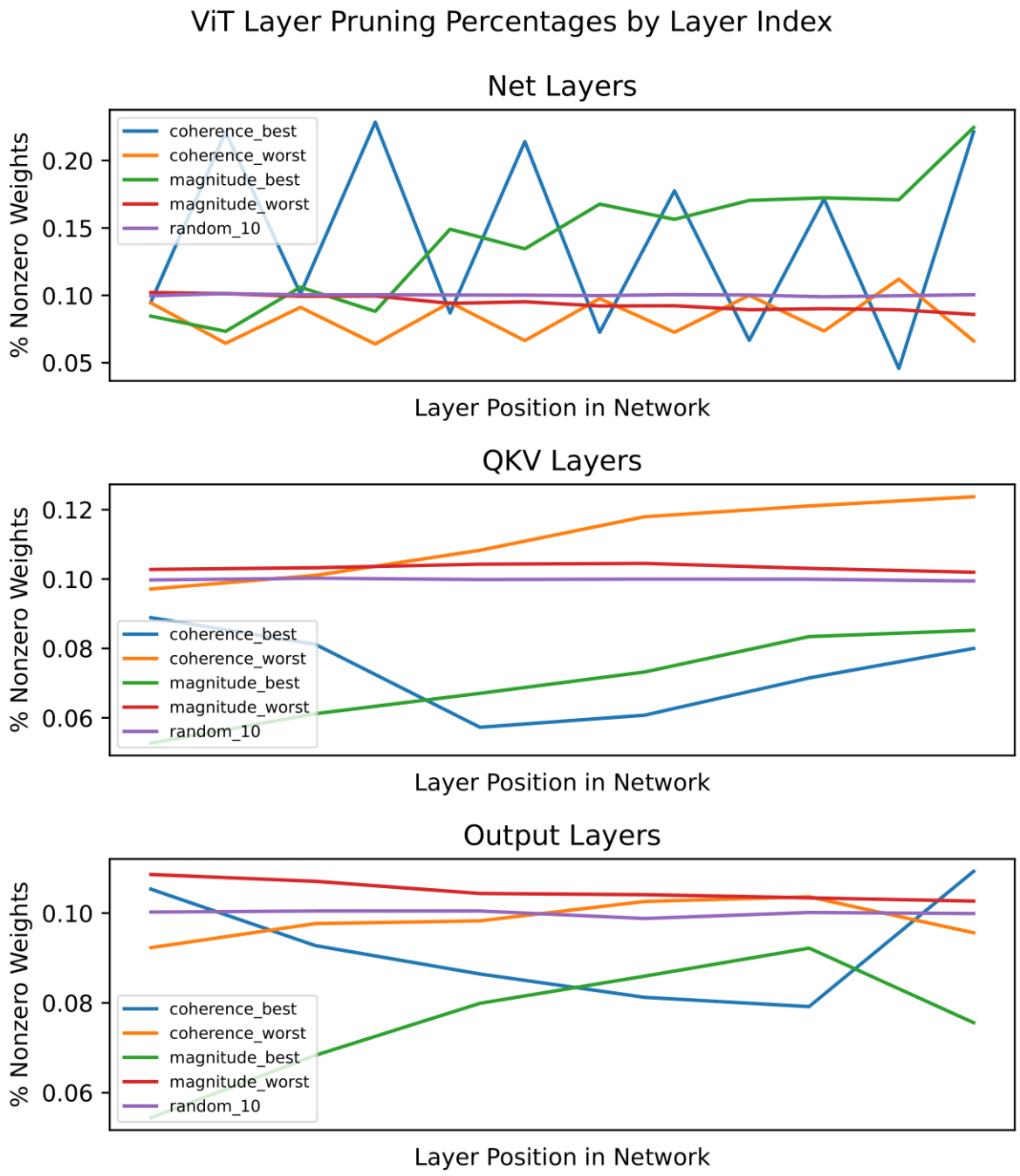

Figure 6.4: Layer-wise sparsity levels resulting from pruning ViT with a variety of methods. ResNet50 layers are divided into three different groups: QKV layers, feedforward hidden layers, and block output layers. The x-axis represents increasing depth within the network. In contrast to Figure 6.3 where some layers were pruned much more than others, ViT layers are pruned far more evenly with no sparsity levels going beyond 5%.

Performance of Locally Pruned ResNet50

Masking Method	Train Accuracy	Test Accuracy
none	0.99974	0.9102
random_10	0.9973	0.8986
magnitude_best	0.99664	0.8901
coherence_best	0.9955	0.8872
magnitude_worst	0.99646	0.887
coherence_worst	0.99722	0.8834

Figure 6.5: Results for pruning ResNet50 to 10% sparsity using various techniques. As opposed to Figure 6.1, layers were pruned uniformly to 10% sparsity, which corrected the poor performance of the "coherence_best" mask.

Performance of Locally Pruned ViT

Masking Method	Train Accuracy	Test Accuracy
none	0.99618	0.7884
coherence_best	0.8494	0.7707
magnitude_worst	0.87538	0.7559
magnitude_best	0.87212	0.7511
random_10	0.87234	0.7461
coherence_worst	0.87292	0.7458

Figure 6.6: Results for pruning ViT to 10% sparsity using various techniques. As opposed to Figure 6.2, layers were pruned uniformly to 10% sparsity

7.1 Analysis

Coherence behaves differently on real and random data, regardless of initialization.

Chatterjee and Zielinski (2022) showed that when a network is trained from a random initialization it will have different levels of diff-based on whether the data was real or random, i.e. whether generalization was occurring. In this paper we showed that not only does this conclusion hold when the network is randomly initialized, it also holds when a network is initialized with fully trained weights and trained on random data. Coherence is not merely an indication of movement in weight space, it seems to be genuinely linked to generalization.

Diffs measured on the same network are distributed differently when coherence is high versus when coherence is low, hinting at a link between diffs and coherence.

Given that large diff magnitudes should theoretically be associated with high coherence, Figures 3.1 and 3.2's illustration of the larger spread of diffs during the max coherence epoch gives empirical evidence for the fact that diffs might somehow align with coherence.

Diffs are not necessarily larger on real data than on random data, calling into question the link between diffs and coherence.

If the connection between diffs and coherence were perfect, then Figures 3.1 and 3.2 would show diffs much larger on real data than on random data. This is not the case. The interpretation of the data most critical to the theory of diffs being an approximation of coherence is that diffs are large when training is occurring because weights have to move in the right direction. When training has slowed down, more weights are already mostly set and so their values oscillate more, leading to low diffs. High coherence seems to coincide with rapid learning, and coherence drops when learning has mostly finished. Therefore diffs and coherence are seem to be largest during the period of rapid learning and small at the end of training, but this is merely a correlation and not a causation. In this scenario, while coherence is not merely a proxy measurement for movement in weight space, diffs are.

Coherence generally increases with the width of a fully-connected network.

Figure 4.3 shows that coherence increases fairly consistently as network width increases. This, however, contradicted by the fact that as coherence grows the generalization gap is also growing. This suggests that increasing coherence may not be a strict indicator of increased generalization.

Just because a network achieves higher max coherence than another network, that doesn't imply that the diffs will be larger

Figure 4.4 shows that even though the width 2800 network has higher max coherence (as seen in Figure 4.3), the diffs are generally narrower than the other. In fact, diff distributions seem to narrow with the width of the network, inversely correlated to coherence. This is another blow to the theory that diffs are linked to coherence.

Pruned networks generalize better than comparably sized dense networks

Figure 5.2 shows that the generalization gap is smaller for the pruned networks, but how are they closing the gap? Do they overfit less at the cost of accuracy, or do they simply achieve better accuracy. Figure 5.1 shows that the pruned networks actually do both. They achieve lower training accuracy and yet higher test accuracy - exactly the results one would expect to see if pruning the network helped to improve generalization.

The performance of sparse networks can be improved without increasing nonzero parameter count, simply by scaling the size of the parent network they are derived from.

Building on the previous point, we discovered that not only do sparse networks perform better than their dense counterparts, but their performance can be improved simply by increasing the size of their dense parent network. In the search for ever more powerful yet compact models, this suggests that it might be possible to expend more resources at training time training a larger dense model and be rewarded with a more powerful model at inference time, without increasing the cost of inference compute.

While diff-based pruning can outperform random pruning, it seems to be

inferior to magnitude pruning for fully-connected and ResNet networks. Diff-based pruning seems to work well on ViT networks.

Looking again at Figures 5.2, 5.1, and 5.3, we see that magnitude pruning generally performed better than diff-based pruning on the fully-connected networks. Figure 6.1 shows that magnitude pruning outperforms diff-based pruning on ResNet as well. Figure 6.2 shows that diff-based pruning performs very well on ViT, to the degree that it is able to match its dense counterpart in performance at a 10% sparsity level.

7.2 Questions Remaining

Why do networks trained on Gaussian data have such large diffs?

Figures 3.1 and 3.2 show that networks trained on random data can still have large diffs. If there is no generalizable pattern to be learned, how can so many samples agree on the direction that parameters should be moving? Shouldn't each gradient be roughly random, meaning that by the law of large numbers the diffs should average out to be near zero?

Why are diffs small when coherence is high

Figure 4.4 shows that even when coherence is high (width 2800) the diffs are narrowly distributed. Our hypothesis is in such an over-parametrized context relatively few parameters are needed to fit the data, and the gradients on the rest of the parameters are more random, meaning the majority of parameters have small diffs. However, this hypothesis needs further investigation.

Why is coherence high when generalization gap is high?

Figure 4.3 shows that coherence increases with the width of a full-connected network, and Figure 4.2 shows that the generalization gap increases monotonically with width. If generalization is meant to increase with width, this forms a contradiction. The performance results show that the test accuracy did not increase with width, only the training accuracy did. With this being the cause of the generalization gap growth, our hypothesis is that the high coherence and high generalization gap have separate causes. The generalization

gap is caused by overfitting due to increased network capacity. The increasing with must be responsible for increased coherence in a way that does not necessarily capture increased generalization. Understanding this relationship requires further investigation.

Why is random pruning so effective, particularly in controlling the generalization gap

In Figure 5.2, the fact that random pruning generalizes so well calls into question the efficacy of other pruning methods. If random pruning does well, are we sure that our other pruning methods are adding much additional value?

The effectiveness of random pruning hints that the effectiveness of our pruning methods is not due to their cleverness but rather due to the inherent properties of large sparse networks. Perhaps our existing pruning methods are not actually choosing the "best" subnetworks, and really any subnetwork will perform reasonably well. Either we don't currently know enough to predict the best subnetworks, or the gap between a really good subnetwork and a random one is rather small.

Magnitude pruning might be good, but it really isn't that much better than magnitude worst, so do we actually understand pruning nearly as well as we would like to think we do? Building on the previous question, tables 6.1, 6.2, 6.5, and 6.6 show that the gap between magnitude pruning and its complete converse (keeping only the weights with the smallest magnitudes) is rather small. If magnitude pruning were really effective at producing especially powerful subnetworks, one should expect this gap to be significantly wider. This "mask inversion problem" needs to be investigated further.

Early magnitude pruning vs late magnitude pruning? How soon can you stop?

In our analysis we noted that diff-based pruning requires information that is available relatively early in training, whereas magnitude pruning requires a fully trained network. We would like to know how soon into training the relative magnitudes of the network's weights is established. That is, at one point in training does the mask derived from magnitude

pruning stabilize? Is it variant throughout all 200 epochs of our training, or does it stabilize maybe 10% of the way through? The answer to this question will determine whether or not diff-based pruning actually saves computational resources over magnitude pruning. If we can compute a magnitude mask after a few epochs of training and get roughly the same mask as one computed at the end of training, we should be doing that.

Why do different networks respond differently to different pruning techniques? Why does ViT respond really well to diff-based pruning when ResNet doesn't? Our results in Chapter 6 show that ResNet50 and ViT respond differently to different pruning methods, both in terms of how well the pruned networks compare to the dense network and in which pruning methods are most effective. In particular, ViT responded very well to diff-based pruning. It is unclear what difference in the architectures is responsible for this difference. Nelson (2022) found that fully-connected networks and CNNs have different "typical ranges" for coherence. Here we showed that different networks respond differently to pruning and coherence.

Does the success of diff-based pruning on ViT mean that it would work well when pruning LLMs?

Given that ViT is a network based on the transformer and it responds very well to diff-based pruning, would LLMs also respond similarly well to diff-based pruning? If so this represents an opportunity to reduce LLM inference costs, which would be a major achievement.

How does pruning scale into the hundreds of billions of parameters

All of our experiments were performed with networks with less than 200M parameters. State of the art LLMs have up to an order of magnitude more parameters and are trained on datasets far more complex than CIFAR-10. How well do these sparsity techniques work at such a scale? If our experiments with fully-connected networks are to be believed, then for any number of parameters a sparse network will outperform a dense network with the same number of parameters. Does this mean that we can generate sparse networks that are

competitive with dense LLMs with the same number of parameters?

7.3 Conclusion

Gradient coherence seems to have some merits as a theory. At the very least, it certainly is a measure that can distinguish between training on real and random data. Diffs as an element-wise approximation of coherence might be a good idea, but the analogy is imperfect as their behaviors don't always match.

In investigating fully-connected networks, we found that increased network width increases coherence. Pruned fully-connected networks perform better than dense ones with the same number of nonzero parameters. Increasing the size of the dense parent network can improve the performance of sparse fully-connected networks.

Pruning also seems to be effective for more complex architectures like ViT and ResNet. While only ViT permitted a pruned model competitive with the dense model, pruned versions of both models were within a few percentage points of parity. In particular, random pruning performed much better than might have been expected.

There are many avenues for further research. For most questions that were answered, several more have arisen in their place. The intersection of generalization and pruning shows promise of being a rich vein to be mined for future progress.

Bibliography

M. Alizadeh, S. A. Tailor, L. M. Zintgraf, J. van Amersfoort, S. Farquhar, N. D. Lane, and Y. Gal. Prospect Pruning: Finding Trainable Weights at Initialization using Meta-Gradients, Apr. 2022. URL http://arxiv.org/abs/2202.08132. arXiv:2202.08132 [cs].

P. Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536, Mar. 1998. ISSN 00189448. doi: 10.1109/18.661502. URL http://ieeexplore.ieee.org/document/661502/.

E. B. Baum and D. Haussler. What Size Net Gives Valid Generalization? *Neural Computation*, 1(1):151–160, Mar. 1989. ISSN 0899-7667, 1530-888X. doi: 10.1162/neco.1989.1.1.151. URL https://direct.mit.edu/neco/article/1/1/151-160/5464.

D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag. What is the State of Neural Network Pruning?, Mar. 2020. URL http://arxiv.org/abs/2003.03033. arXiv:2003.03033 [cs, stat].

S. Chatterjee and P. Zielinski. On the Generalization Mystery in Deep Learning, June 2022. URL http://arxiv.org/abs/2203.10036. arXiv:2203.10036 [cs].

T. Dettmers and L. Zettlemoyer. Sparse Networks from Scratch: Faster Training without Losing Performance, Aug. 2019. URL http://arxiv.org/abs/1907.04840. arXiv:1907.04840 [cs, stat].

U. Evci, T. Gale, J. Menick, P. S. Castro, and E. Elsen. Rigging the Lottery: Making All Tickets Winners, July 2021. URL http://arxiv.org/abs/1911.11134. arXiv:1911.11134 [cs, stat].

J. Frankle and M. Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks, Mar. 2019. URL http://arxiv.org/abs/1803.03635. arXiv:1803.03635 [cs].

J. Frankle, G. K. Dziugaite, D. M. Roy, and M. Carbin. Stabilizing the Lottery Ticket Hypothesis, July 2020. URL http://arxiv.org/abs/1903.01611. arXiv:1903.01611 [cs, stat].

J. Frankle, G. K. Dziugaite, D. M. Roy, and M. Carbin. Pruning Neural Networks at Initialization: Why are We Missing the Mark?, Mar. 2021. URL http://arxiv.org/abs/2009.08576. arXiv:2009.08576 [cs, stat].

T. Gale, E. Elsen, and S. Hooker. The State of Sparsity in Deep Neural Networks, Feb. 2019. URL http://arxiv.org/abs/1902.09574. arXiv:1902.09574 [cs, stat].

M. Glasgow, C. Wei, M. Wootters, and T. Ma. Max-Margin Works while Large Margin Fails: Generalization without Uniform Convergence, Mar. 2023. URL http://arxiv.org/abs/2206.07892. arXiv:2206.07892 [cs].

S. Hochreiter and J. Schmidhuber. Flat Minima. *Neural Computation*, 9(1):1–42, Jan. 1997. ISSN 0899-7667, 1530-888X. doi: 10.1162/neco.1997.9.1.1. URL https://direct.mit.edu/neco/article/9/1/1-42/6027.

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, Feb. 2017. URL http://arxiv.org/abs/1609.04836. arXiv:1609.04836 [cs, math].

N. Lee, T. Ajanthan, and P. H. S. Torr. SNIP: Single-shot Network Pruning based on Connection Sensitivity, Feb. 2019. URL http://arxiv.org/abs/1810.02340. arXiv:1810.02340 [cs].

P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz. Importance Estimation for Neural Network Pruning, June 2019. URL http://arxiv.org/abs/1906.10771. arXiv:1906.10771 [cs, stat].

V. Nagarajan and J. Z. Kolter. Uniform convergence may be unable to explain generalization in deep learning, Oct. 2021. URL http://arxiv.org/abs/1902.04742. arXiv:1902.04742 [cs, stat].

M. V. Nelson. Gradient Conditioning in Deep Neural Networks, 2022.

B. Neyshabur, R. Tomioka, and N. Srebro. In Search of the Real Inductive Bias: On the Role of Implicit Regularization in Deep Learning, Apr. 2015. URL http://arxiv.org/abs/1412.6614. arXiv:1412.6614 [cs, stat].

B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. Exploring Generalization in Deep Learning, July 2017. URL http://arxiv.org/abs/1706.08947. arXiv:1706.08947 [cs].

M. Sabanayagam, F. Behrens, U. Adomaityte, and A. Dawid. Unveiling the Hessian's Connection to the Decision Boundary, June 2023. URL http://arxiv.org/abs/2306.07104. arXiv:2306.07104 [cond-mat, stat].

K. Sreenivasan, J.-y. Sohn, L. Yang, M. Grinde, A. Nagle, H. Wang, E. Xing, K. Lee, and D. Papailiopoulos. Rare Gems: Finding Lottery Tickets at Initialization, June 2022. URL http://arxiv.org/abs/2202.12002. arXiv:2202.12002 [cs].

H. Tanaka, D. Kunin, D. L. K. Yamins, and S. Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow, Nov. 2020. URL http://arxiv.org/abs/2006.05467. arXiv:2006.05467 [cond-mat, q-bio, stat].

C. Wang, G. Zhang, and R. Grosse. Picking Winning Tickets Before Training by Preserving Gradient Flow, Aug. 2020. URL http://arxiv.org/abs/2002.07376. arXiv:2002.07376 [cs, stat].

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization, Feb. 2017. URL http://arxiv.org/abs/1611.03530. arXiv:1611.03530 [cs].

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, Mar. 2021. ISSN 0001-0782, 1557-7317. doi: 10.1145/3446776. URL https://dl.acm.org/doi/10.1145/3446776.