CATCHING THE LONG TAIL IN DEEP NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning dynamics in deep neural networks are still a subject of debate. In particular, the identification of eventual differences regarding how deep models learn from frequent versus rare training examples is still an area of active research. In this work, we focus on studying the dynamics of memorization in deep neural networks, where we understand memorization as the process of learning from rare or unusual training examples that are part of the long-tail of a dataset. As a working hypothesis, we speculate that during learning some weights focus on mining patterns from frequent examples while others are in charge of memorizing rare long-tail samples. Using this idea, we develop a method for uncovering which weights focus on mining frequent patterns and which ones focus on memorization. Following previous studies, we empirically verify that deep neural networks learn frequent patterns first and then focus on memorizing long-tail examples. Furthermore, our results show that during training a small proportion of the total weights present an early convergence to model frequent patterns, while the vast majority of the weights present a slow convergence to model long-tail examples. We also find that memorization happens mostly at the first layers of a network and not at the level of classification. Finally, by analyzing performance differences for models trained with varying levels of long-tail samples, we find that a larger number of long-tail samples has a negative impact on learning frequent patterns, by a process we conjecture to force the model to learn frequent patterns as memorization.

1 INTRODUCTION

Understanding the learning dynamics of deep neural networks is still a challenge. There are many factors to consider: architecture, datasets, optimization algorithms, etc. One notable fact of Deep Neural Networks (DNNs) is their capacity for predicting elements in the long-tail of a distribution. We define the long-tail as the rare or unusual examples in a distribution. A recent theory (Feldman, 2020) has postulated that this capacity for extrapolation is due to memorization of long tail elements in the training distribution. In this work, we seek to study how these elements are learned and how their training dynamics differ from those of frequent elements. Recent work suggests that networks have two separate modes of learning: first, a phase that learns most easy-to-find patterns; afterwards, a second phase that memorizes most elements from the long tail.

As a first goal, we focus on validating that, indeed, these two phases exist. To achieve this, we first derive a strategy to identify training examples that are related to frequent and long tail sets. Afterward, we analyze the learning dynamics of the network's weights associated to both sets. After this, we run experiments to validate if the common idea that there are separate weights in a network devoted to pattern learning and memorization has any basis to it, and if so, whether these weights can be localized.

Our method is based on ideas presented in several previous works. First, we take inspiration from Arpit et al. (2017) which states that DNNs learn patterns first, noise later. We also consider a main thesis from Javed et al. (2020) which states that weights associated with non-spurious features show smaller variance than spurious weights. Considering these two works, we hypothesize that if there are specialized weights for long-tail samples, they will converge later in training than weights that learn frequent patterns. Furthermore, our analysis takes inspiration from Arpit et al. (2017) and Zhang et al. (2017), which analyze training dynamics of DNNs under different levels of noise.

Similarly, we also focus our analysis in visual recognition problems. Furthermore, in our datasets, we artificially manipulate the long tail by simulating infrequent examples by sampling instances from unused categories.

In terms of the identification of the set of weights that are associated to learning frequent and long tail training instances, we first conduct an experiment to validate not only the existence of the two-phase learning scheme, but also to identify a suitable transition point between these two phases. Afterwards, by comparing model weights at the point where the change of phase occurs with model weights at the end of training, we can associate weights that remain mostly the same as related to pattern learning and those that change as related to long-tail memorization.

Thus, our main contributions can be summarized as:

- We provide evidence for the phenomenon that DNNs allocate different set of weights to learn frequent and long-tail samples. We also find evidence that this learning happens in two phases: mining frequent patterns and memorizing long-tail instances, and we derive a simple method to identify the point where the change of phases occurs.
- We develop a method to find *where* memorization is taking place in a network. Specifically, we find evidence that memorization does not concentrate in the classification layers, as it was pointed out by Feldman & Zhang (2020). Furthermore, we show which layers involved in representation learning do most of the memorization.
- We find that training with longer-tailed datasets seems to reduce the efficiency of learning natural patterns, as it forces models to learn frequent patterns as if they were being memorized. As a relevant observation, we find that a nontrivial amount of weights remain mostly the same throughout the entire training procedure.

Section 2 will explain our methodology, while Section 3 will introduce our experimental setup. Section 4 will provide discussion of the results of those experiments. Section 5 will review related work. Section 6 will do a summary of our conclusions.

2 METHODOLOGY

Our overall goal is to understand the learning dynamics of neural networks. As a first step, we focus on validating if there are two phases in Deep Neural Network training: pattern learning and memorization. Then, we focus on finding where this phase change occurs. Furthermore, we also study if we can identify network weights that are related to pattern learning and long-tail memorization.

2.1 FINDING EVIDENCE FOR THE TWO PHASES AND A TRANSITION POINT

In order to validate the existence of the two phases mentioned above, during learning, we directly try to identify a transition point that maximizes the difference in accuracy between the following sets:

- **Frequent:** a fixed subset of images of the training set which feature frequent patterns. To select images from the set of frequent patterns, we use the *Agreement* metric from Carlini et al. (2019). This metric ranks the training examples accordingly to its uniqueness. In particular, we select examples from the 80% of examples with lowest rank, i.e., the more common examples. This selection follows the long-tail distribution (Zhu et al., 2014), where few natural sub-categories of elements represent a large part of the dataset, which we consider as frequent examples.
- **Long Tail:** a fixed subset of images of the training set which features *artificial* long-tailed data.

Following our analysis, we experimentally discover a point where the first derivative of the accuracy for the frequent set is equal to the first derivative of the accuracy of the long-tail set. Thus, we hypothesize that this is an appropriate point to define the end of a first phase of learning frequent samples and the start of the long-tail learning phase. As our estimation of the accuracy is done coarsely over each epoch, we will also perform a sensitivity analysis around this point to validate our hypothesis. We dub this point the "Phase Change Point" (PCP).

2.2 Localization of Weights

Starting from the PCP, we study the convergence of weights over the remaining training epochs to determine which weights are related to learning frequent patterns and which ones are related to memorization. This idea comes from a result in Arpit et al. (2017), where they find that DNNs learn patterns first, noise later. Thus, we would expect convergence of weights associated with pattern learning to happen earlier than for memorization weights. Using this idea, to identify each set of weights, we will keep a copy of the model at the PCP, then proceed to train it until it achieves maximum performance in both sets. Once the training is over, whatever weights remain *roughly the same* after the end of training should be related to learning patterns. Those that do not, should be related to learning long-tail samples.

We define the informal *'roughly the same'* as follows. First, we perform L2 normalization for both convolutional and linear layers. We find this to be necessary as we observed that the final classification layer increases in scale during training, while its weights maintain, for the most part, their ordering. Then, we take the absolute value of the difference of the weights and compare it with a threshold t. t is simply the mean of the absolute value of a weight for a given layer multiplied by an arbitrary constant $\alpha \in [0.05, 0.1, 0.2]$. In practice, we found $\alpha = 0.1$ to be a good value for our analysis. Notation-wise, we define weights that stay the same (with regards to the PCP) during training, *convergent*, while those that do not will be called *non-convergent*.

To test our hypothesis, we will purge weights above the threshold in two different ways:

- **Keep:** we set the weights to their value at the PCP. This is akin to what is usually done for some Lottery Ticket (Frankle & Carbin, 2019) derived methods for reinitializing a network, however, we do this only for weights that are above the threshold.
- **Purge:** we set the weights to 0.0.

3 Experiments

For all experiments, we use as a baseline the model at the PCP. The models are then trained for additional epochs up to 50. For the specific PCPs used for each model, please see the Appendix. All hyperparameters used are detailed in the Appendix as well.

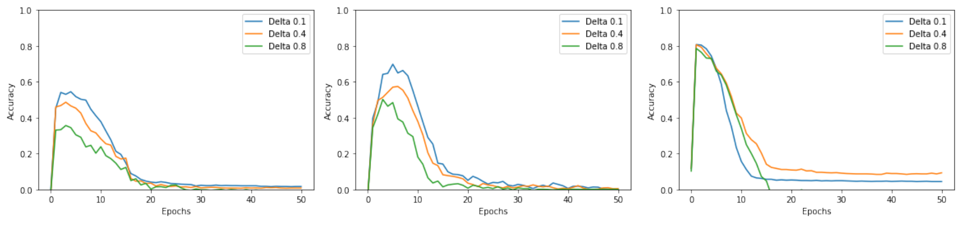
All our models and experiments are coded in PyTorch (Paszke et al., 2019) and will be made available publicly on acceptance. We train a reduced version of AlexNet (Krizhevsky et al., 2012) as used in Arpit et al. (2017) and Zhang et al. (2017). This is a CNN with 2 convolutional layers (with 5x5 and 200 filters each) and 3 linear layers (384, 192, 10). We refer to this network as MiniAlexNet in this work. We also train on a Resnet-18 (He et al., 2016) to see if our results apply to deeper models.

All experiments are run on CIFAR-10 (Krizhevsky et al., a) and MNIST (LeCun & Cortes, 2010). Also, in the spirit of Zhang et al. (2017) and Arpit et al. (2017), we add long-tail samples to our training data. We add an extra 0% to 80% of the training set as long-tail samples. Our long-tail samples are created by assigning random elements from random classes of CIFAR-100 (Krizhevsky et al., b) when using CIFAR-10 and by shuffling pixels from MNIST samples when using MNIST. We do this to modulate the amount of long tail our model needs to learn. These two ways of generating long-tail samples allow us to also distinguish between a long tail that has rare patterns (CIFAR-100) and another one that has none (shuffled MNIST).

4 Results

4.1 Phase Change Analysis

We analyze accuracy evolution for the Frequent and Long Tail sets. In particular, we plot the curve of the difference between these two values, as can be seen in Figure 1. Where this curve achieves its maximum is where we expect a Phase Change to occur, as it would imply that the first derivative of the Frequent set goes below the growing rate of the Long Tail set. We know that the first derivative of the Frequent set has to be larger at the start than for the Long Tail, because of Arpit et al. (2017) results and our own experiments. From that point onward, *learning long-tail samples should dominate learning*. The amount of long-tail samples seems to reduce the magnitude of these curves, but

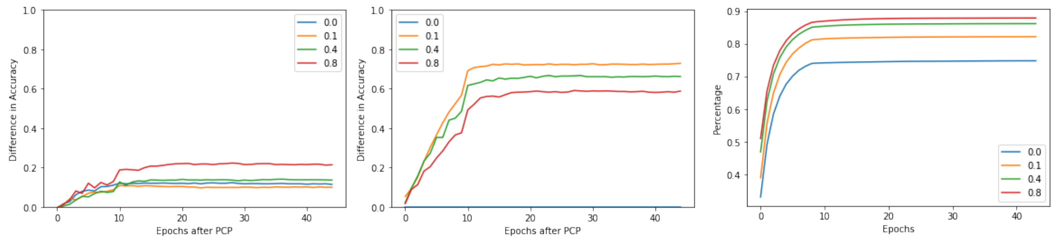


(a) CIFAR10 - Mini-Alexnet (b) CIFAR10 - Resnet-18 (c) MNIST - Mini-Alexnet

Figure 1: Delta of accuracy over training epochs between the *Frequent* and *Long Tail* for varying amounts of long-tail samples. We see a clear maxima happen for each model and dataset. These maxima are the points where first derivatives for both sets match and is where we define the phase change between learning frequent patterns and learning long-tail elements.

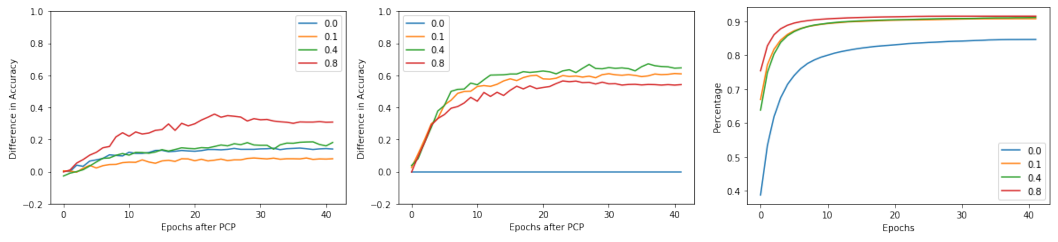
does not seem to move the point where the maximum happens. On the other hand, dataset complexity and model architecture seem to matter. For a deeper model, such as Resnet18, we see that the peak is achieved later than the model MiniAlexNet. And for the same model, we see that a simpler dataset such as MNIST displaces this peak to earlier moments during the training process. In the following subsection, we show how these points act as a watershed on what is being learned.

4.2 Analysis by Weight Convergence



(a) Frequent Set (b) Long Tail Set (c) % of Non-Convergent Weights

Figure 2: Difference in accuracy over remaining training epochs between the regularly trained model and models where *'Keep'* was applied to all layers for MiniAlexNet - CIFAR-10. We can see that the impact on the Frequent set is moderate, while for the Long Tail Set the impact on accuracy is dramatic.



(a) Frequent Set (b) Long Tail Set (c) % of Non-Convergent Weights

Figure 3: Difference in accuracy over remaining training epochs between the regularly trained model and models where *'Keep'* was applied to all layers for Resnet18 - CIFAR-10. We can see that the impact on the Frequent set is greater than with MiniAlexNet, yet the impact on the Long Tail set is still more acute. More depth seems to increase the interference effect of long-tail samples on pattern mining weights.

Now that we have candidates for the PCP, we begin by analyzing the impact of weight convergence across training. We observe (Figures 2c, 3c) that a greater amount of long-tail correlates with a greater amount of non-convergent weights for all architectures. As expected, we observe that for datasets with greater amounts of long-tail, more weights are required to be adapted. The same happens when we increase dataset complexity between MNIST and CIFAR-10, as CIFAR-10 seems to require more weight adaptation than MNIST.

We also checked what happened when weights during training were compared to their *initialization* weights. We find that still a significant proportion of weights (between 10% and 40%) remain mostly the same during training. Their role is not explored in this work.

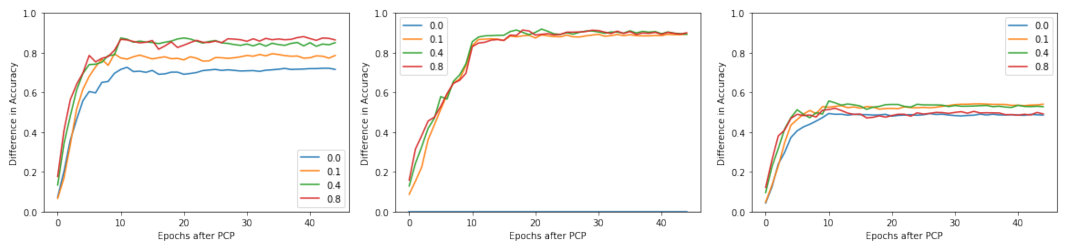
4.3 Effects of Non-Convergence on Performance

We follow by understanding the importance of non-convergent weights on accuracy on the Frequent and Long Tail sets. What we see (Figure 2) is that returning non-convergent weights to their original value - that is using *Keep*- has a disproportionate effect on Long Tail samples. This shows as an absolute performance drop of 60-70%, while a much smaller drop on the Frequent set (20%). Thus, we can conclude that whatever training is happening on those weights has a more direct impact on memorization than on learned patterns.

We see a more significant drop in performance on the Frequent set when we see a larger amount of long-tail samples. This leads us to conclude that when training with more long-tail samples, these affect the learning of frequent patterns and act as a sort of *interference* for Frequent sample learning. As the proportion of long-tail samples increases, the typical training batch will contain increasing proportions of long-tail to frequent samples. What might have been considered a frequent sample before, later might be thought of as a rarer example that needs memorization. Thus, learning is affected because it is not as efficient to be optimizing our loss for the frequent patterns as it was before.

Using *Purge*, however, dramatically lowers performance for all sets involved as we can see in Figure 4. We hypothesize this has to do with the network adapting to these weights' starting configuration and building its solution from that point. We speculate these weights to be related to the weights pruned on the Iterative Pruning steps of the Lottery Ticket Hypothesis, while a subset of those we keep is the Lottery Ticket itself. We will not pursue this idea further in this work.

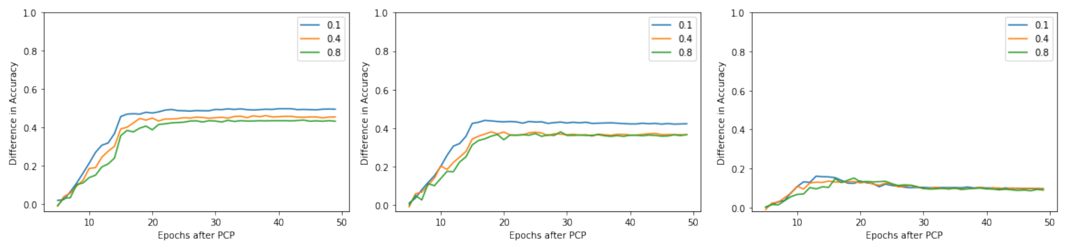
We would like to point out that when applying *Keep* to a deeper model such as Resnet-18 (Figure 3), we see that for increased levels of long-tail samples, this gap between performance in the Frequent and Long Tail sets diminishes. Compared to MiniAlexNet, Resnet-18 has much more capacity and is much deeper. Thus, we can conclude that Resnet-18 is learning patterns in a way that is more akin to memorization than pattern-learning. We believe this result supports the claims in Zhang et al. (2020) that deeper networks seem to be doing more memorization than shallower structures. Because with increased long-tail samples, pattern-mining weights seem to explain less and less of the Frequent set. This is interesting because it is usually thought that memorization happens mostly in the final classification layers. In the next subsection, we show evidence that this is not the case.



(a) Frequent Set (b) Long Tail Set (c) Test Set

Figure 4: Difference in accuracy over remaining training epochs between the regularly trained model and models where *'Purge'* was applied to all layers for MiniAlexNet on CIFAR-10. We can see that the impact is drastic for all sets, not only for the Long Tail. We hypothesize this has to do with the model optimizing itself *in spite of these weights*.

4.4 Effects of Weight Dissimilarity by Layer



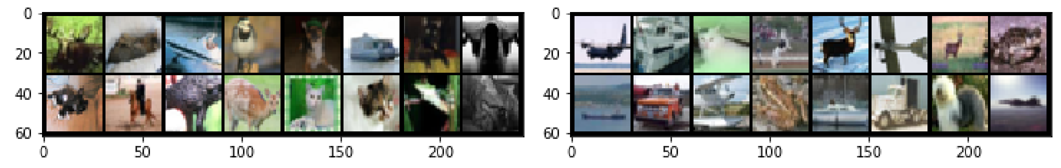
(a) First 2 Layers (b) 3rd Layer (c) Last 2 Layers

Figure 5: Difference in accuracy in the Long Tail set over remaining training epochs between the regularly trained model and models where *'Keep'* was applied to a selection of layers for MiniAlexNet on CIFAR-10. We can see the effect of restoring weights in the first layers has much more impact on accuracy than in the final classifier layers as would be expected.

We selectively disable layers using *Keep* to understand which layers seem to affect memorization the most. Contrary to the usual intuition, the classification layers seem not to be the primary source of memorization in the network. We find that a lot of memorization happens in the earlier layers that build the representation. This agrees with results from Section 3.6 of Feldman & Zhang (2020), however, we are also able to identify specific layers where this memorization has the most impact. As we can see in Figure 5, using *Keep* on the Convolutional Layers of our MiniAlexNet network has the most drastic impact on performance on the Long Tail set.

Given the sequential nature of these models, it is expected that changes closer to the input might have more consequences downstream. Nevertheless, the total impact of using *Keep* on the Long Tail set is around 60-70%, while the impact of just altering the last two layers is around 15-30% at the most. Thus there's still a significant gap in accuracy that is explained by the representation learning layers.

Anecdotally, while classification layers may not be the most relevant part of a model for memorization, we did find that for one of our models this layer seemed to hurt generalization performance, while earlier layers did nothing of the sort. This may be an interesting avenue for research, however we do not pursue it further in this work.



(a) Incorrectly Classified - Frequent Set - CIFAR-10 (b) Correctly Classified - Frequent Set - CIFAR-10

Figure 6: Samples from the Frequent set classified after applying Keep. To the left are samples that were classified incorrectly; to the right, samples correctly classified. Incorrectly classified samples show harder examples, usually closeups or with an atypical color palette. To the right, we see easier examples where we get a full view of the class being predicted (car, plane).

5 Related Work

5.1 Memorization

Most studies of noise in DNNs have to do with overcoming its impact or using it to improve network performance, as is the case of Dropout (Srivastava et al., 2014). Zhang et al. (2017) showed that DNNs are able to learn arbitrary noise given enough parameters. Arpit et al. (2017) qualify this assertion by showing that while networks can indeed learn noise, training dynamics for noise and natural patterns are markedly different: noise takes more work to learn than natural patterns. We arrive at the same conclusion but characterize this via the point at where weights converge, while also being able to pinpoint where this is happening in the network.

Previous work (Feldman, 2020) has postulated the hypothesis that memorization of long tail samples in the training set is what is behind the phenomenal success of Deep Learning in extrapolating to unseen long tail samples. In followup work (Feldman & Zhang, 2020), a measure of memorization is developed for individual samples based on the probability of being correctly classified when the model is trained with or without it. These metrics also show that memorization does not happen at the classification layers but is part of the representation. We find evidence to support this, and we also quantify where and how much each layer contributes to the memorization effort. In the same vein, Carlini et al. (2019) developed a set of metrics to understand how representative of their class is a specific sample. They base their analysis on robustness to adversarial attacks, how well a model handles not having the sample in training, and both the agreement and confidence of models in classifying a particular sample. We use their *agreement* metric to select the most common elements of our datasets to construct our Frequent sets.

Zhang et al. (2020) have studied how modulating the depth of DNNs affects their memorization capabilities when faced with the extreme example of learning just one sample: greater depths seem to make DNNs degenerate to learning a constant function. In other words, the network is forgetting the input and encoding its weights.

Javed et al. (2020) studies how to identify spurious features -which are usually related to memorization- in the online learning setting. Based on the idea -taken from Arjovsky et al. (2020)- that causal features should have stable performance across different environments, they develop a method to eliminate spurious features. They speculate that spurious features in this setting will tend to have greater variance, which they apply to their method with great success. This work was the inspiration behind the weight convergence method used for analysis used in the present work.

5.2 Phases of Neural Network Training

The Information Bottleneck method (Tishby & Zaslavsky, 2015) poses the problem of representation learning as an optimization problem in two mutual information terms: $I(X, Z)$ and $I(Z, Y)$; the first representing compression and the second one sufficiency (i.e. having enough information to solve the task). They find that Deep Learning training is subject to two phases: fitting and compression (Shwartz-Ziv & Tishby, 2017). In the first, the sufficiency term is maximized, while the compression term is minimized in the second. Compression also acts as a means of forgetting the input, which may suggest the memorization of specific samples in the weights of the network.

Another view of how learning happens in DNNs comes from the Lottery Ticket Hypothesis (Frankle & Carbin, 2019). In simple terms, it states that there are subnetworks within an untrained DNN that have won the ”initialization lottery”. That is, their weights are especially susceptible to SGD optimization finding a good solution. Iterative pruning of networks reveals the ”winning ticket” which matches or surpasses performance of the original network. Successive works (Zhou et al., 2019) show that masks can be learned over the weights of an untrained network to achieve nontrivial performance. Taking this to the extreme, Ramanujan et al. (2020) have developed the *edge-popup* algorithm to find subnetworks in untrained DNNs with accuracy that matches supervised methods.

6 Conclusions

We have run experiments to test the hypothesis that weights specialize in learning patterns or in memorization and that there are two phases of learning in Deep Neural Networks. We have found evidence for both of these ideas: we have found that the division between the two phases can be found by finding the point where accuracy curves for Frequent and Long Tail sets achieve the same first derivatives; we have also found that returning weights to their value at the Phase Change Point, has a disproportionate adverse effect on the performance of long-tail samples versus natural samples, which suggests that whatever is being learned after that point has to do mostly with memorization.

We have studied what proportion of weights are involved in each type of learning as well. We find that most of the weights are devoted to memorization while a relative minority are involved in mining frequent patterns. This agrees with the results of the Lottery Ticket Hypothesis, which tells us that a minority of weights can explain most of the generalization performance of a network.

A relevant contribution of our work is the idea that weights dedicated to memorization can be localized within a network. Thus, we have identified that while some memorization happens in the classification layer of a Deep Neural Network, most of the memorization is taking place in earlier layers. Furthermore, we find that the greater the amount of long-tail samples, the more interference the network will experience when trying to learn natural samples. This interference manifests itself as a reduction of the efficiency of learning natural samples which we think is due to forcing models to learn frequent patterns as if they were instances to be memorized.

As future work, there are still many questions to answer: turning these non-convergent weights completely off reduces performance drastically in both long-tail and natural samples. What is the role of these weights? Is there a relationship between these behaviour and the Lottery Ticket Hypothesis? Also, why are so many weights essentially the same from start to finish? Do they add anything significant to training or are they simply unwanted neighbours of the training landscape?

REFERENCES

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *stat*, 1050:27, 2020.

Devansh Arpit, Stanislaw K Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron C Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *ICML*, 2017.

Nicholas Carlini, Úlfar Erlingsson, and Nicolas Papernot. Distribution density, tails, and outliers in machine learning: Metrics and applications. *arXiv preprint arXiv:1910.13427*, 2019.

Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pp. 954–959, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450369794. doi: 10.1145/3357713.3384290. URL https://doi.org/10.1145/3357713.3384290.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *arXiv preprint arXiv:2008.03703*, 2020.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Khurram Javed, Martha White, and Yoshua Bengio. Learning Causal Models Online. 2020. URL http://arxiv.org/abs/2006.07461.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). a. URL http://www.cs.toronto.edu/~kriz/cifar.html.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research). b. URL http://www.cs.toronto.edu/~kriz/cifar.html.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11893–11902, 2020.

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *CoRR*, abs/1703.00810, 2017. URL `http://arxiv.org/abs/1703.00810`.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014. URL `http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf`.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pp. 1–5. IEEE, 2015.

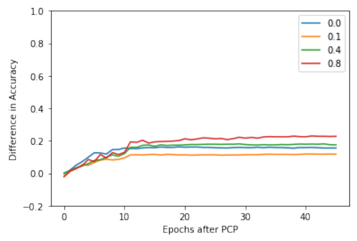
Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL `https://openreview.net/forum?id=Sy8gdB9xx`.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Michael C. Mozer, and Yoram Singer. Identity crisis: Memorization and generalization under extreme overparameterization. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=B1l6y0VFPr`.

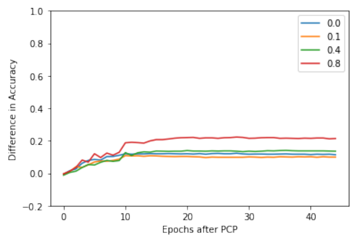
Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *Advances in Neural Information Processing Systems*, pp. 3597–3607, 2019.

Xiangxin Zhu, Dragomir Anguelov, and Deva Ramanan. Capturing long-tail distributions of object subcategories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 915–922, 2014.

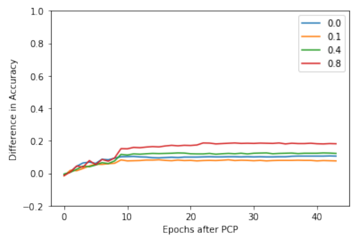
A Sensitivity Analysis over Phase Change Point - Frequent Set



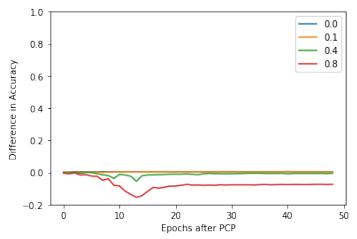
(a) Epoch 4 - MiniAlexNet - CIFAR10



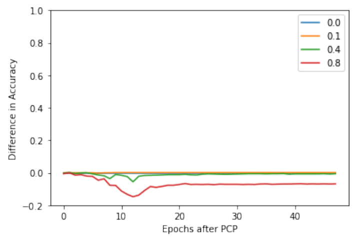
(b) Epoch 5 - MiniAlexNet - CIFAR10



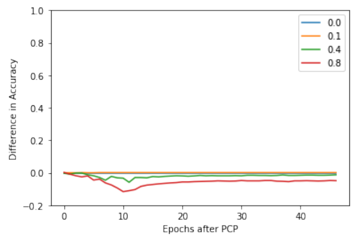
(c) Epoch 6 - MiniAlexNet - CIFAR10



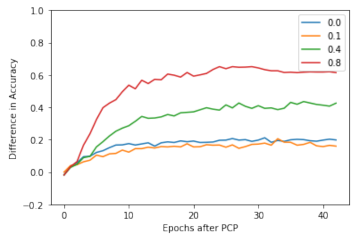
(d) Epoch 1 - MiniAlexNet - MNIST



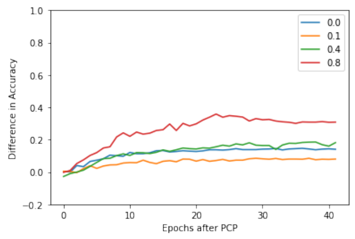
(e) Epoch 2 - MiniAlexNet - MNIST



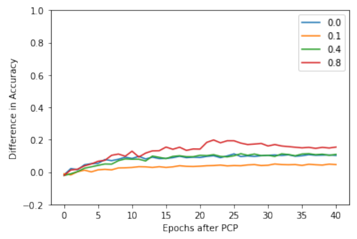
(f) Epoch 3 - MiniAlexNet - MNIST



(g) Epoch 7 - Resnet-18 - CIFAR10



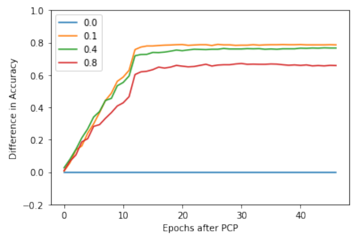
(h) Epoch 8 - Resnet-18 - CIFAR10



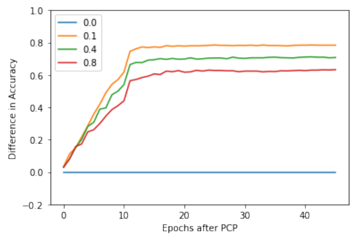
(i) Epoch 9 - Resnet-18 - CIFAR10

Figure 7: Difference in Accuracy of applying *Keep* for different Phase Change Points on the Frequent Set. The earlier the PCP, the higher the differences. PCPs are 5, 1, 8 for MiniAlexNet - CIFAR10, MiniAlexNet - MNIST and Resnet-18 - CIFAR-10 respectively.

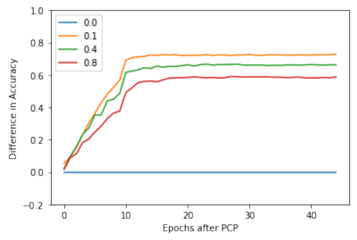
B Sensitivity Analysis over Phase Change Point - Long Tail Set



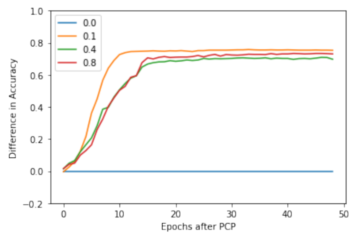
(a) Epoch 4 - MiniAlexNet - CIFAR10



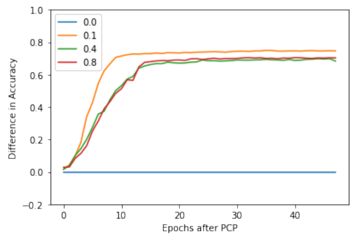
(b) Epoch 5 - MiniAlexNet - CIFAR10



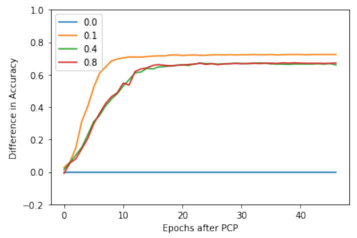
(c) Epoch 6 - MiniAlexNet - CIFAR10



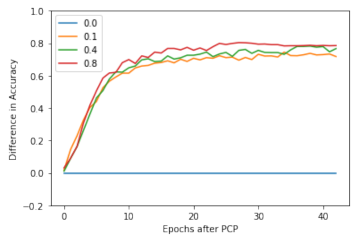
(d) Epoch 1 - MiniAlexNet - MNIST



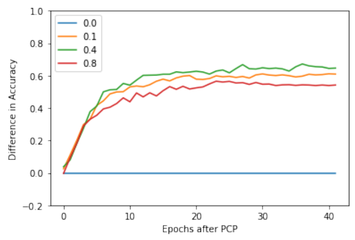
(e) Epoch 2 - MiniAlexNet - MNIST



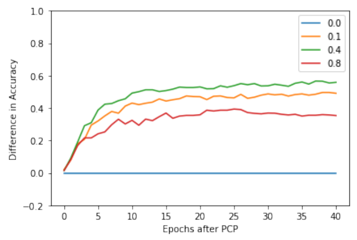
(f) Epoch 3 - MiniAlexNet - MNIST



(g) Epoch 7 - Resnet-18 - CIFAR10



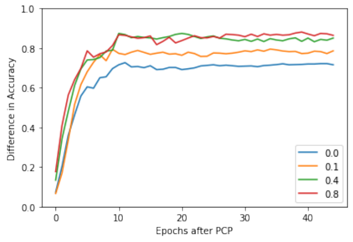
(h) Epoch 8 - Resnet-18 - CIFAR10



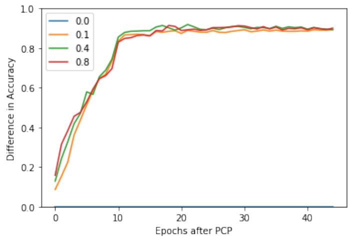
(i) Epoch 9 - Resnet-18 - CIFAR10

Figure 8: Difference in Accuracy of applying *Keep* for different Phase Change Points on the Long Tail Set. The earlier the PCP, the higher the differences. PCPs are 5, 1, 8 for MiniAlexNet - CIFAR10, MiniAlexNet - MNIST and Resnet-18 - CIFAR-10 respectively.

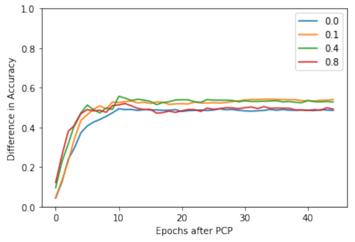
C Impact of Purge on different Models



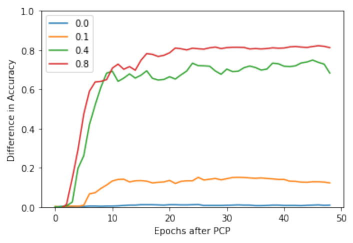
(a) Frequent - AlexNet - CIFAR10



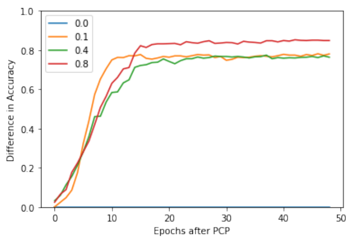
(b) Long Tail- AlexNet - CIFAR10



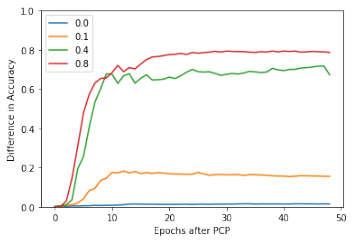
(c) Test- AlexNet - CIFAR10



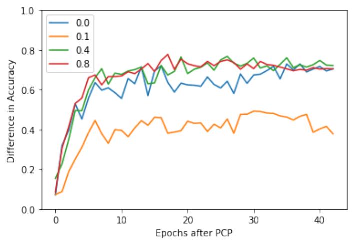
(d) Frequent - AlexNet - MNIST



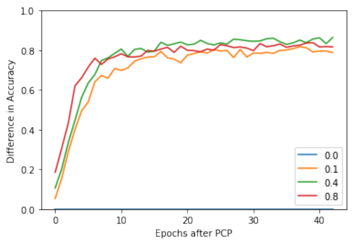
(e) Long Tail- AlexNet -MNIST



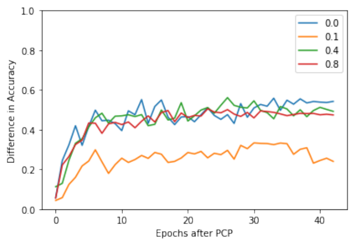
(f) Test- AlexNet - MNIST



(g) Frequent - Resnet-18- MNIST



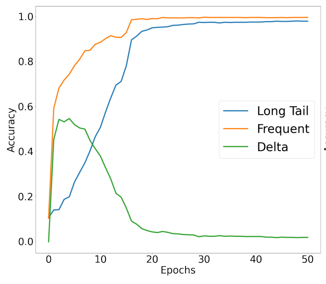
(h) Long Tail - Resnet-18 -MNIST



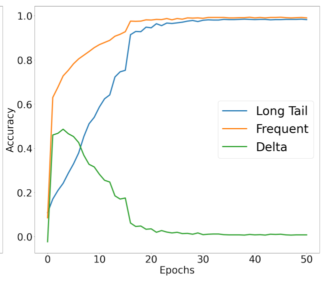
(i) Test - Resnet-18 - CIFAR10

Figure 9: Difference in accuracy in the Long Tail set over remaining training epochs between the regularly trained model and models where *'Purge'* was applied to all layers starting from the PCP. We can see drastic impact on all sets.

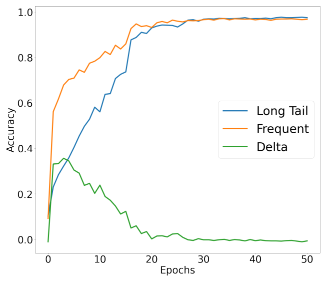
D Phase Change Points with Accuracies



(a) CIFAR10-Mini-Alexnet-10%

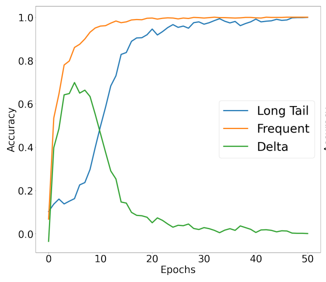


(b) CIFAR10-Mini-Alexnet-40%

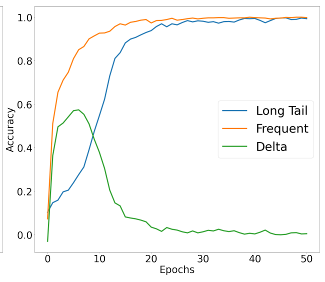


(c) CIFAR10-Mini-Alexnet-80%

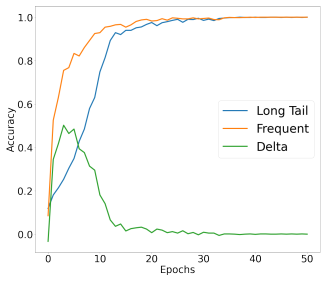
Figure 10: Accuracy obtained by both *Frequent* and *Lont Tail* sets in CIFAR10 in the Mini-Alexnet model. The *Delta* curve shows the difference between the accuracy obtained in both sets.



(a) CIFAR10-Resnet-18-10%

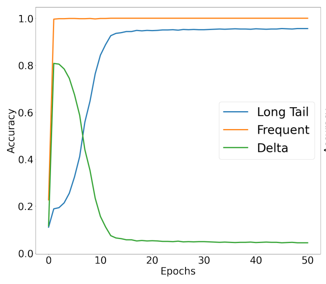


(b) CIFAR10-Resnet-18-40%

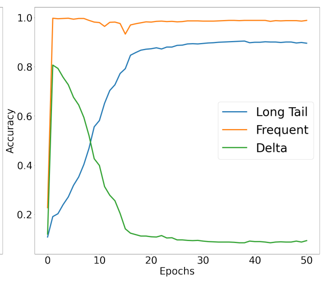


(c) CIFAR10-Resnet-18-80%

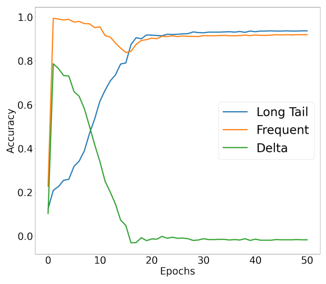
Figure 11: Accuracy obtained by both *Frequent* and *Lont Tail* sets in CIFAR10 in the Resnet-18 model. The *Delta* curve shows the difference between the accuracy obtained in both sets.



(a) MNIST-Mini-Alexnet-10%



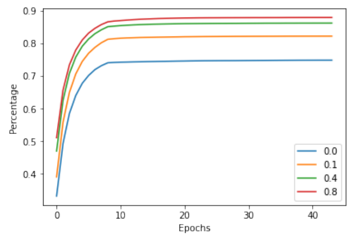
(b) MNIST-Mini-Alexnet-40%



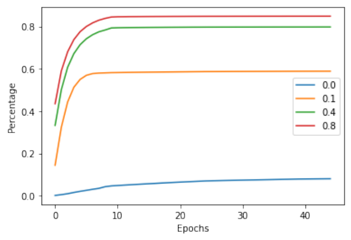
(c) MNIST-Mini-Alexnet-80%

Figure 12: Accuracy obtained by both *Frequent* and *Lont Tail* sets in MNIST in the Mini-Alexnet model. The *Delta* curve shows the difference between the accuracy obtained in both sets.

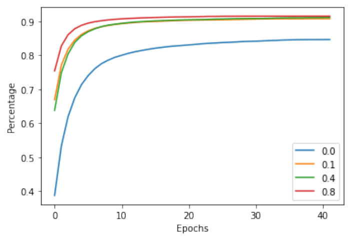
E Proportion of Non-Convergent Weights for All Models



(a) MiniAlexNet - CIFAR10

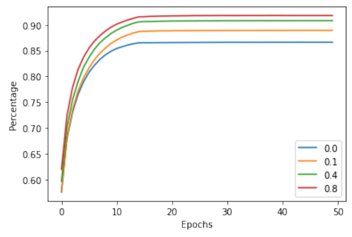


(b) MiniAlexNet - MNIST

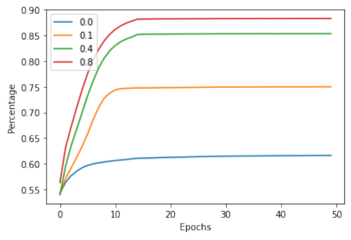


(c) Resnet-18 - CIFAR10

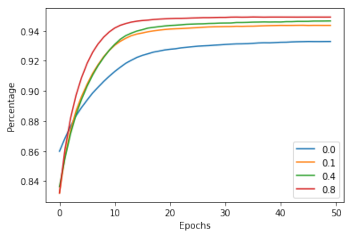
Figure 13: Proportion of parameters that are non-convergent relative to the Phase Change Point for different amount of long tail examples and models. Increased long-tail samples show much more non-convergent parameters, which suggests greater use of parameters. We relate convergent parameters to those that learn frequent patterns.



(a) MiniAlexNet - CIFAR10



(b) MiniAlexNet - MNIST



(c) Resnet-18 - CIFAR10

Figure 14: Proportion of parameters that are non-convergent relative to the initialization point for different levels of long-tail samples and models. Increased amount of long-tail samples show much more non-convergent parameters, which suggests greater use of parameters.

F Training Hyperparameters

We train our models using SGD with Momentum 0.9. No scheduler is used during the whole of training.

Table 1: Training Hyperparameters

Model	**Dataset**	**Optimizer**	**LR**	**Batch Size**	**Momentum**
MiniAlexNet	MNIST	SGD	0.01	512	0.9
MiniAlexNet	CIFAR-10	SGD	0.01	512	0.9
ResNet-18	CIFAR-10	SGD	0.1	128	0.9

G PHASE CHANGE POINTS

In this section we list the Phase Change Points used for analysis for each model trained. These are based on the point where the difference between the Frequent and Long Tail set is maximum. However, as this curve is far from smooth, we do a sensitivity analysis around this point to find an appropriate point for analysis. For models trained with no long-tail samples we will use the PCP associated to the same model trained with 10% of long-tail samples.

Table 2: Phase Change Points for different models trained.

Model	**Dataset**	**Long-Tail (%)**	**Phase Change Point (Epoch)**
MiniAlexNet	MNIST	0.0	1
MiniAlexNet	MNIST	10.0	1
MiniAlexNet	MNIST	40.0	1
MiniAlexNet	MNIST	80.0	1
MiniAlexNet	CIFAR-10	0.0	5
MiniAlexNet	CIFAR-10	10.0	5
MiniAlexNet	CIFAR-10	40.0	5
MiniAlexNet	CIFAR-10	80.0	5
ResNet-18	CIFAR-10	0.0	8
ResNet-18	CIFAR-10	10.0	8
ResNet-18	CIFAR-10	40.0	8
ResNet-18	CIFAR-10	80.0	8