Exploring Lazy Training in Deep Feedforward Neural Networks

Madeleine Kearns (mk4652), Kenneth Munyuza (km3829), Sam Brosh (sdb2174)

Spring 2023

Abstract

Deep neural networks have revolutionized machine learning and are gaining significant press for their potential applications on a wide range of tasks. In recent work, a 'lazy training' regime has been documented, in which the weights of a heavily over-parameterized network do not move significantly during training but remain close to their random initialization. This allows the network to avoid overfitting and reduces the heavy computational cost traditionally associated with training deep neural networks. In this project, we confirm the existence of lazy training through experiments with random synthetic data and establish a level of significance for weight movement to classify when the lazy training regime is in effect. This allows us to determine how over-parameterized a network must be for lazy training to occur. Furthermore, we explore applying lazy training to networks deeper than one hidden layer, on both the synthetic data and the MNIST dataset.

1 Introduction

Deep learning is a sub-field of machine learning that has received wide acclaim for its high performance in diverse fields, covering image processing [1], computer vision, natural language processing [2], recommendation systems, and many more. It attracts much interest due to its goal of mimicking human decision-making processes. It consists of

training artificial neural networks to learn complex data patterns and features in a manner that allows them great flexibility to cope with high-dimensional and unstructured data. Networks typically consist of multiple layers of interconnected nodes that process the data and make predictions, with the connections between nodes being weighted during training through back-propagation to minimize the training error. Despite its widespread success, there remain many facets of the functioning of neural networks that are poorly understood. One such area is the fact that even over-parameterized neural networks, where the number of features in the model exceeds the number of observations in the training data, do not tend to overfit and can still produce an impressive performance on unseen data [3][4][5][6][7]. This research area is complicated by the inherently low interpretability of neural networks [8].

Research into over-parameterized neural networks led to the observation of a phenomenon wherein heavily over-parameterized networks, approaching infinite width, did not have their randomly initialized weights move during the training process and yet were still achieving high performance on unseen data. This was dubbed lazy training. Specifically, under lazy training, strongly over-parameterized networks trained with gradient-based methods can converge linearly to zero training loss without their weights moving from their random initializations. There has been discussion about whether this lazy training may be the reason for the high performance of neural networks in high-dimensional spaces where traditional machine learning techniques struggle; it has been largely decided that lazy training is actually due to the algorithms inherently scaling the network and therefore intrinsically select parameters close to the initialization. However, lazy training remains an interesting regime for the training of neural networks, especially moving away from infinite-width networks which are rarely realistic. In this project, we investigate how over-parameterized a neural network needs to be before lazy training occurs, starting with a network that is completely characterized by an equal number of training observations and neurons, and incrementally widening the network until lazy training sets in. This experiment is run on synthetic data so that there is no underlying

structure for the network to learn, requiring it to completely fit the training data.

Current literature solely focuses on the analysis of lazy training on shallow neural networks with one hidden layer, and it is accepted that for these networks to exhibit lazy training, the hidden layer needs to approach infinite width. However, infinite-width neural networks are rarely implementable in practice due to their excessive number of neurons, and in other subfields of neural network research, there has been much work focused on the benefits and trade-offs of reducing network width and increasing depth. Although these networks are not theoretically well-understood, they are widely used in practice, and thus experimentation with them to increase understanding is of great interest. In general, deeper and narrower networks can learn more complex and abstract representations of the input data but are challenged by vanishing/exploding gradients leading to convergence problems, while shallower and wider networks can model more complex functions (single hidden layer, infinitely wide neural networks are universal approximators [9][10][11]) by are prone to overfitting and poor generalization. To further research lazy training, we look at increasing the number of layers in the network while maintaining a lazy training regime, investigating how much narrower a deeper network can be. Although the theory of applying lazy training to deep neural networks is not yet well-understood, we hope this analysis can be useful for establishing its behavior past one hidden layer.

We apply the same network structure to the MNIST dataset, establishing that the lazy training regime is maintained. Through this, we aim to maintain focus on the neural network architecture and the impact of changes to it, rather than becoming overconcerned with the dataset. We also investigate the behaviour of neural networks of more than 1 hidden layer on the MNIST dataset.

This work is experimental, and although we will provide a high-level overview of the networks and algorithms used, the analysis will be focused on the additional understand-

ing that the results give to the field of lazy training and neural network interpretability.

2 Background and Related Work

Many theoretical works have shown that it is possible for over-parameterized neural networks to converge to zero training loss with their parameters varying either very small amounts or not at all, through the use of gradient-based optimization algorithms [12][13][14]. This has been dubbed lazy training [15]. This has been determined to be an implicit bias phenomenon, wherein the model behaves like a linearization around its (random) initialization [15]. Intrinsic to this behavior is the width of the network; it has been proven that the width must be approached infinite for the parameters to stay static around the initialization, as with the increases in width, the training causes increasingly smaller changes to the parameters proportional to the parameters [12][16][17][18]. It is shown in [19] that infinitely wide neural networks behave like the linearisation of the network around its random initialization, and this is the behavior that we are hoping to define a limit to in this work. The behavior described here has been accepted to be due to an implicit choice of scaling, where the introduction of a lazy training regime necessitates the introduction of an implicit scaling factor. This is discussed in [15], where an explicit scaling factor is used to show that any parametric model can be trained with and enjoy the benefits of the lazy regime if, at initialization, the output is close to zero.

Another fascinating aspect of the lazy training regime is the connection it provides between neural networks and kernel methods. This was posited in [20], where a family of kernel functions was defined that can mimic shallow and deep neural network architectures and proves the link between neural networks trained with gradient descent methods and kernel methods. The Neural Tangent Kernel (NTK) was proposed in [19], and it has been shown that NTKs can get zero training loss when the network is sufficiently over-parameterized, with the parameters not moving significantly from their initializations – just as in lazy training in neural networks [19][14][21]. With the NTK

regime, it is possible to draw a parallel between training neural networks with gradient descent optimization strategies with solving kernel regression [15]. Therefore, NTKs can be said to be equivalent to a lazy training regime where the initialization is on a very large scale, restricting the neurons to stay local to the initialization. Some work has gone into providing empirical performance estimates of infinite width NTK applications, with [17] giving an algorithm for the computation of the NTK regime on convolutional neural networks.

In other related work, it has been suggested that even neural networks that are only mildly over-parameterized can achieve zero training loss when the data is generated using a teacher neural network and initial loss is lower than a polynomial threshold [22]. Furthermore, with regards to our later work in extending the application of lazy training to networks of more than one hidden layer, we are trying to work alongside the study completed in [23][24][25] where it is shown that deep networks may be more difficult to train due to the necessity of nonlinear optimization operations caused by the ..., but can learn more complex mappings that shallow networks as their inputs are passed through several layers of nonlinear processing.

We chose to extend our analysis to the MNIST dataset due to its previously established reputation as a benchmark dataset for neural networks. It is established that networks where the layers are randomly initialized, and where only the top layer has gradient descent applied in training (also referred to as weakly trained neural networks) have a good performance on the MNIST dataset [17]. By using a common benchmarking dataset, we can ensure that the network architecture is being evaluated in a way that allows for fair and objective comparisons.

It has also been established that deep neural networks can express functions much more efficiently compared to single layer networks, reducing the overall number of neurons needed by decreasing the necessary width [26][27]. We aim to combine this result with the lazy training phenomenon by investigating the onset of lazy training in deep

neural networks, and in each of their individual layers.

3 Problem Formulation

There has been a series of theoretical works that show that an overparameterized, randomly initialized, infinitely wide, single-layer neural network can have its weights move very little during training and yet still converge linearly to zero training loss. This phenomenon has been dubbed *lazy training*. [15] present the lazy training regime largely as follows. Note: they use R for the loss function, while here we use L as in the lecture slides.

Consider a parameter space $\mathbb{R}^p$, a Hilbert Space $\mathcal{F}$, and a smooth model h. Assume that loss is smooth $L : \mathcal{F} \to \mathbb{R}_+$. The aim is to use gradient-based methods as in Equation 1 to minimize the objective function $F : \mathbb{R}^p \to \mathbb{R}_+$, defined as in Equation 2.

$$\omega_1 = \omega_0 - \eta \nabla F(\omega_0) \tag{1}$$

$$F(\omega) := L(h(\omega)) \tag{2}$$

For any initialisation ω_0 that is part of the parameter space $\mathbb{R}^p$, the resulting model (linearised around the initialisation) is notated as in Equation 3 with the objective function in Equation 4.

$$\overline{h}(\omega) = h(\omega_0) + Dh(\omega_0)(\omega - \omega_0) \tag{3}$$

$$\overline{F}(\omega) L(\overline{h}(\omega)) \tag{4}$$

At the beginning of the training process, the optimization paths of F and $\overline{F}$ starting from ω_0 are naturally close together. Lazy training is the phenomenon where these paths remain close or identical throughout the training process. Next, we will identify when lazy training occurs. As a definition, lazy training refers to situations where

the differential of the model h does not significantly change, while the loss decreases significantly. We define the relative changes of the objective and differential in Equations 5 and 6 respectively.

$$\Delta(F) := \frac{|F(\omega_1) - F(\omega_0)|}{F(\omega_0)} \tag{5}$$

$$\Delta(Df) := \frac{\|Df(\omega_1) - Df(\omega_0)\|}{\|Df(\omega_0)\|} \tag{6}$$

For lazy training to occur, $\Delta(F) >> \Delta(Dh)$ as shown in Equation 7. We assume that ω_0 is not a minimizer or a critical point so that $F(\omega_0) > 0$ and $\nabla F(\omega_0) \neq 0$

$$\frac{|F(\omega_1) - F(\omega_0)|}{F(\omega_0)} >> \frac{\|Df(\omega_1) - Df(\omega_0)\|}{\|Df(\omega_0)\|} \tag{7}$$

The gradient descent algorithm from Equation 1 gives Equations 8 and 9 the linearizations around the initial parameters.

$$F(\omega_1) \approx F(\omega_0) + (\omega_1 - \omega_0) \cdot \nabla F(\omega_0) = F(\omega_0) - \eta \|\nabla F(\omega_0)\|^2 \tag{8}$$

$$Df(\omega_1) \approx Df(\omega_0) + (\omega_1 - \omega_0) \cdot D^2 f(\omega_1) = Df(\omega_0) - \eta \nabla F(\omega_0) \cdot D^2 f(\omega_1) \tag{9}$$

Using these, Equation 7 can be simplified as shown in Equation 10.

$$\begin{aligned} \Delta(F) &\approx \eta \frac{\|\nabla F(\omega_0)\|}{F(\omega_0)} \\ \Delta(Df) &\approx \eta \frac{\|\nabla F(\omega_0)\| \left\|D^2 f(\omega_0)\right\|}{\|Df(\omega_0)\|} \\ \Delta(F) &>> \Delta(Df) \\ \implies \frac{\|\nabla F(\omega_0)\|}{F(\omega_0)} &>> \frac{\left\|D^2 f(\omega_0)\right\|}{\|Df(\omega_0)\|} \end{aligned} \tag{10}$$

Df is the Gradient/Jacobian $Df(\omega(t)) = (\nabla f(x_1; \omega(t)), ..., \nabla f(x_1; \omega(t)))^T \in \mathbb{R}^{n \times p}$ and $D^2 f$ is a Hessian matrix.

The condition in Equation 10 simplifies based on the choice of the loss function to give the relative scale of the model at ω_0 using the approximation in Equation 11.

$$\|\nabla F(\omega_0)\| = \left\|Df(\omega_0)^T(f(\omega_0) - y)\right\| \approx \|Df(\omega_0)\| \, \|f(\omega_0) - y\| \tag{11}$$

In this project, we will use this theory of lazy training to explore networks that train in the lazy regime even though they are not infinitely wide. In these networks, a large α is equivalent to the large variance in the initialization of the weight parameters ω_0. We will start by examining a one-hidden-layer neural network. This can be characterized as in Equation 12. [15] proves that as long as $\alpha m \to \infty$, which is the case for any $\alpha > 0$ and large m, all shallow networks are guaranteed to reach the lazy regime.

$$\begin{aligned} f(x) &= \alpha(m) \sum_{j=1}^{m} \alpha_j \sigma(\omega_j \cdot x) \\ &=: \alpha(m) \sum_{j=1}^{m} \phi(\theta_j, x) \end{aligned} \tag{12}$$

The Neural Tangent Kernel is defined in **??**, where $\nabla_\omega f$ represents the path of steepest descent for quadratic loss.

$$\Theta(\omega, x_1, x_2) = K(x_1, x_2) = \nabla_\omega f(\omega, x_1) \cdot \nabla_\omega f(\omega, x_2) \tag{13}$$

For infinite-width networks, the NTK is deterministic, although for finite-width networks it is random. 8 and 9 show the linearizations of the model around the initial parameters; this is also called the Tangent Model and, once combined with an assumption of quadratic loss, this gives the neural tangent kernel shown in Equation 13.

4 Reproduction

In order to establish the lazy training regime, a simple two-layer network was trained on synthetic data using PyTorch. The training data was randomly generated and had 100 observations with 10 features drawn from a normal distribution with a mean of 0 and a standard deviation of 1. The data were randomly labeled with an equal number of observations in each binary class. The number of neurons in the hidden layer of the

network was initially equal to the number of observations in the training data, in order to ensure that the network was completely characterized, and then incrementally increased in steps of 10. Lazy training has traditionally been understood to occur in infinite width, so, to mimic this behavior, we examined the behavior of networks with up to two orders of magnitude more neurons than the training data had observations.

For each network, the initial and final weights were saved and the Frobenius Norm between them was calculated. The norm was used instead of the absolute difference as it is a more precise way of measuring the difference between the two, and tends to result in lower variance. The results for each network were also normalized with respect to the number of neurons to avoid penalizing the larger networks. The loss function used was Binary Cross-Entropy, and the optimization function used was the Adam algorithm. This algorithm builds on stochastic gradient descent to make it more efficient through the use of adaptive learning rates and momentum-based gradient descent. After the initial and final weights for each network were obtained, the percentage difference between the two was calculated and plotted, showing a clear decrease in movement as the network became ever more over-parameterized. This completed our reproduction of the past results, establishing that lazy training occurs as networks become more over-parameterized and cause the weights of the network not to move significantly around their random initializations.

To validate these results on real-world data, we apply a similar training method to the MNIST dataset, scaling the network parameters to fully- and overly-characterize this data. Figures XX and XX show the plots for the synthetic and MNIST data respectively and validate previous commentary on lazy training being a specific training regime that sets in at high levels of over-parameterization.

5 Methods

There were several steps that we needed to employ in order to analyze the properties of finite-width, multi-layer networks. First, we needed to lay down some design considerations so that our experiments were unbiased and displayed data that was truly characteristic of the neural networks that we were testing. This was especially important when comparing neural networks with different numbers of layers.

The first consideration was to keep the number of computations constant between the different neural networks we were testing. During testing, we were faced with a decision on whether to keep the number of neurons constant or to keep the number of weights constant. When we first ran our experiments, we kept the number of neurons the same. However, when we viewed the results, we found data that was inconsistent with our hypothesis: the multi-layer neural network seemed to be converging on the infinite width theorem faster than the single-layer neural network. This is likely due to the fact that in deeper fully connected networks, there are many more parameters than in single-layer networks.

Therefore, we chose to ensure that the sum of the elements in each weight matrix was equivalent across each neural network that we tested. For example, when comparing one-layer neural networks to two-layer neural networks, we utilized a weight matrix of

$$dim(W_1) = d \times n$$

for the one layer neural network, whereas for the two-layer neural network, we used weight matrices of

$$dim(W_1) = d \times \frac{n}{2}$$

and

$$dim(W_2) = \frac{n}{2} \times \frac{n}{20}$$

where d is the dimensionality of the observations, and n is the number of neurons used

in the one layer network. The important part here is that number of computations for both cases are both dn. This ensures that the deeper networks are not given an unfair advantage against shallower networks, as it has been shown that the number of weights in a network are the key to its expressivity.

Another important design consideration was to ensure that the prior distribution under the weights during each initialization was the same across the experiments. The default prior distribution for weight initialization in the TensorFlow library is glorot uniform, which is variable depending on the weight matrix associated with the dense layer. In order to ensure that each experiment was fair across networks of different sizes and layers, we utilized the following uniform distribution throughout the entirety of our experiments:

$$w_i \sim U(-0.05, 0.05)$$

.

We deemed it important that each network was both sufficiently and equivalently overparameterized to the 100 samples of ten-dimensional data. However, since we faced restrictions on the computational power and time available, we were forced to compromise on the accuracy and precision of the experiments. We chose a factor ε to train our neural network until it surpassed that loss (i.e. early stopping). This way, we could complete our experiments in a reasonable amount of time, while producing results with an acceptable level of accuracy that we could make interesting observations and assertions on.

Lastly, we aimed to examine neural network lazy training behavior on structured data. The MNIST dataset is a widely used benchmark dataset in machine learning and computer vision. It has widely been used to evaluate the performance of neural networks due to its simplicity and interpretability. It consists of a collection of 70,000 gray-scale

images of handwritten digits (0 to 9), each of size 28 by 28 pixels. The training data is used for this work and consists of 60,000 images. This dataset is especially useful for us because we wanted to test our hypothesis that lazy training can be exhibited despite the use of data that contains structure. However, as we had computational power and time constraints, we had to employ transfer learning as a technique to decrease the total time of the experiments. We utilized the neural network shown in Figure 1.

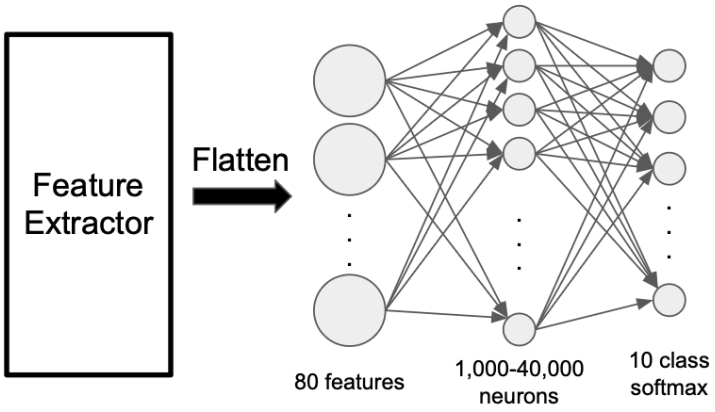
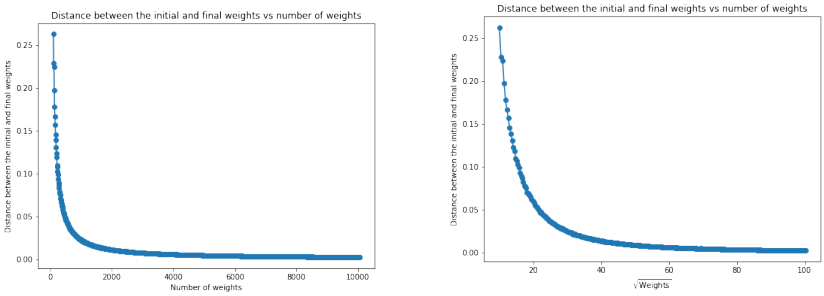


Figure 1: Single Layer Distance Plot

To pretrain the feature extractor, we used a concatenation of a custom feature extractor followed by a 1,000-length fully connected layer of neurons, followed by a 10-class softmax layer. The feature extractor model is based on EfficientNetB0 architecture and was trained until 99.0% accuracy was achieved. We reshaped the input sizes of the handwritten digits from 28x28 pixel images to 32x32 so that they could be input into the architecture. We then turned off the trainability of the pre-trained feature extractor and modified the top of the model into variable-sized neurons from 1,000 to 40,000 to test if the lazy training phenomenon is exhibited on datasets with structure. Finally, we trained each neuron set for 5 epochs, or until the model reached about 99.6% accuracy to ensure that the model was overfitting.

6 Results & Discussion

After reproducing previous work validating the existence and onset of lazy training, we compared Figures 2 and 3 to demonstrate that there is an appropriate boundary for the amount of weight movement that could still be considered to be operating within the lazy training regime.



(a) standard

(b) rescaled using Geiger *et al.* method

Figure 2: Single Layer Distance Plot

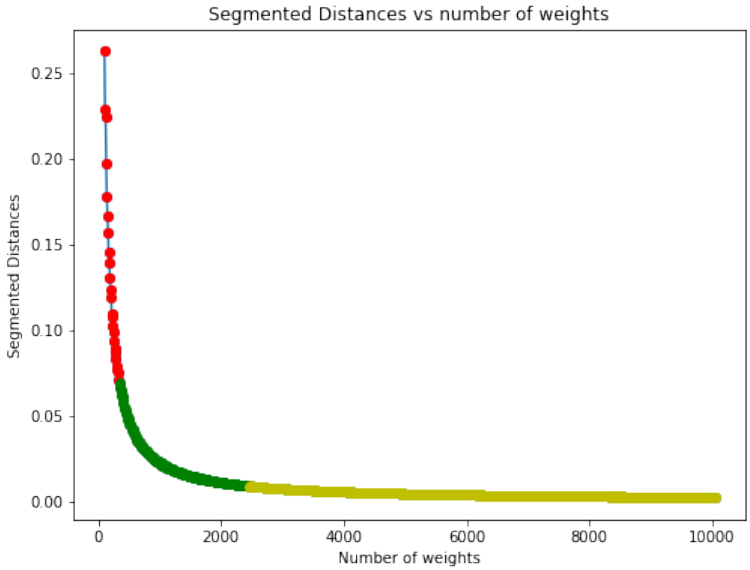
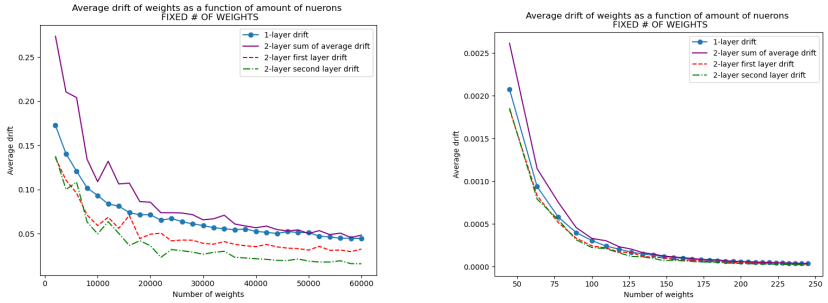


Figure 3: Segmented Single Layer Distance Plot

To then extend the analysis to networks with more diverse architectures, we began to examine the possibility of characterizing the lazy training bound for deeper, narrower networks. We investigated networks with two hidden layers, as well as investigating the individual behavior of each layer in the network.



(a) One vs. two layer, unscaled

(b) One vs. two layer, rescaled using Geiger *et al.* method

Figure 4: A comparison of one and two-layer movement of neurons

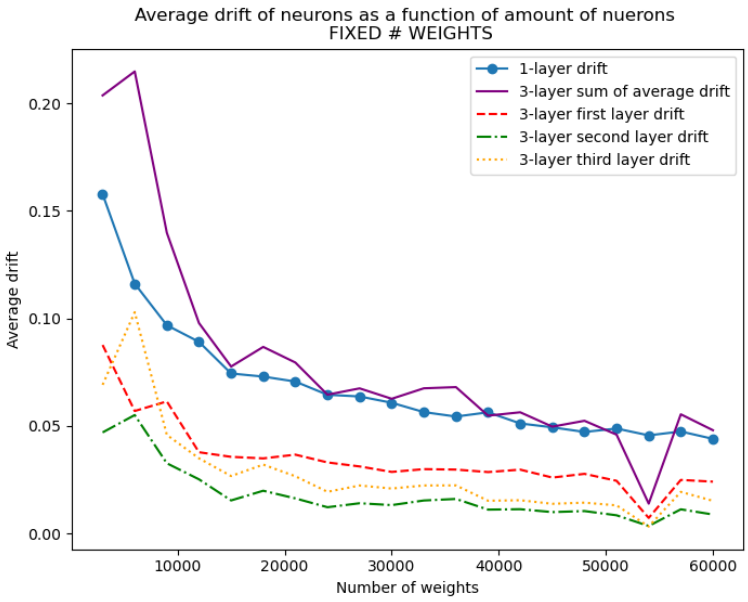


Figure 5: A comparison of one and three-layer movement of neurons

As suggested by the worsened performance of the network with two hidden layers on

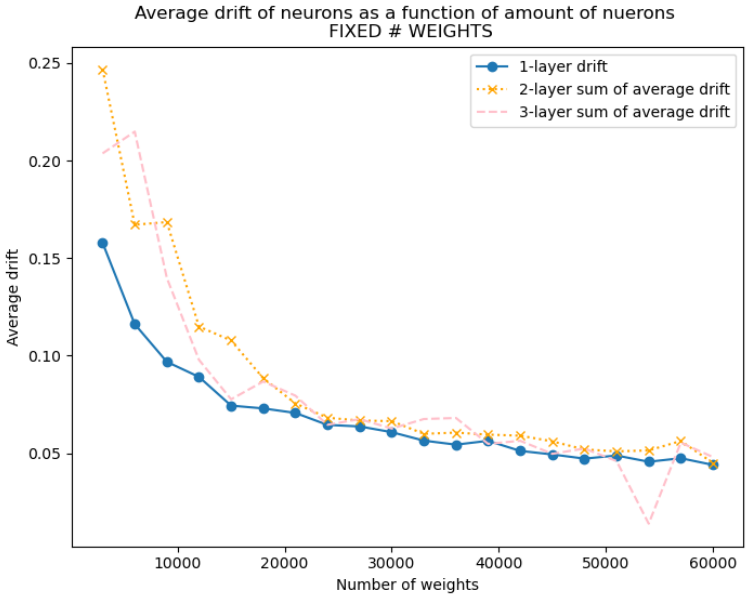
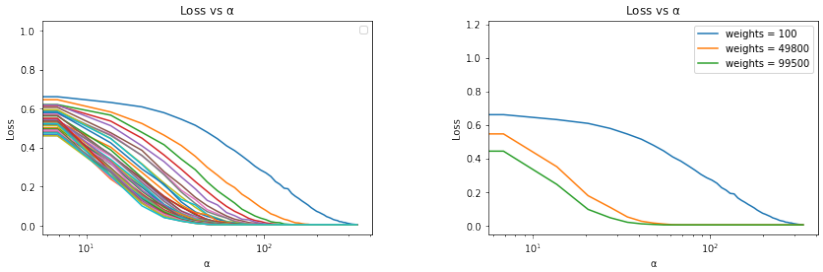


Figure 6: A comparison of one, two, and three-layer movement of neurons

the synthetic data, shown in Figure 4, networks with more than one hidden layer are not guaranteed to be universal approximators. This differentiates them from networks with only one hidden layer, which has been widely demonstrated to be universal approximators [9] [10] [11]. This is because the complexity of the function that a neural network can approximate is determined by the composition of its activation functions, and the composition of activation functions in a two-layer neural network is more limited than in a network with one hidden layer. Specifically, a two-layer neural network can only approximate functions that are separable, meaning that the decision boundary between different classes can be defined by a linear combination of the input features. This is not the case with the randomly generated synthetic data used here. We continued our study with an analysis of three-layer networks. As seen in Figure 6, it was interesting to see that despite the odd network structure (weight matrices of $(d \times n/3)$, $n/3 \times n/30$, $n/30 \times n/3$) we were required to employ to keep the number of computations the same, we saw similar or faster convergence to the infinite width theorem than the two-layer network. This further reinforces the assertion we made previously regarding the two-layer neural network's lack of separability.

Going back to our previous implementation of lazy training on our synthetic data with a single hidden layer Neural Network, we mapped the behavior of our loss during this process to gain further inferences on its corresponding behavior. Binary Cross-entropy loss was used as our loss criterion given the binary nature of our task. It calculates the discrepancy between the target variable's actual probability distribution and the projected probability distribution as seen in equation 14. We used it to stop in finite type given our over-parametrized lazy training regime as seen in figure 7. You can see the obvious shift in the loss in the plotted logarithmic scale as the behavior approaches what is expected in the infinite width theorem from its initial small width in the leftmost curve to the widest tested network at the rightmost curve.

$$\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i) \tag{14}$$



(a) loss vs α in single-layer network

(b) loss vs α labeled

Figure 7: loss movement as changes in weight occur in our single-layer network with synthetic data

We then opened up our scope from neural networks with a completely random prior distribution to the prior distribution of handwritten digits on a sheet of paper contained in the MNIST dataset. This proves that our results hold for networks operating on datasets containing more underlying structures than the random synthetic data previously used.

We see the neural networks that are overfitting to this dataset are exhibiting a lazy

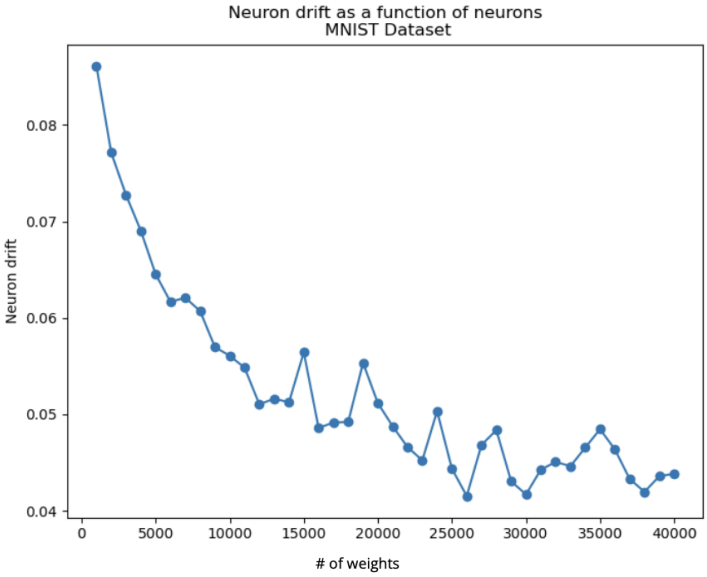


Figure 8: Weight movement as a function of the number of weights

training regime as we increase the total number of computations. This experiment can stand as the groundwork for future investigation into neural networks as research departs the purely theoretical, synthetic realm, and enters an era where we are more focused on results relating to real-world data. As an extension, we wanted to investigate the differences between this example and synthetic data by setting up a new experiment where we classified 100 data points from a synthetic dataset in which each data point had 1,280 dimensions, or the same number of features as the training data had. However, the attempts proved futile because the complexity of the data as well as the labels proved to be too irregular for a neural network to reach any sort of progress in classification, and thus training. Therefore, we could not complete this experiment. However, for future work, this would be an interesting comparison to make so that some relationships can be formed.

7 Conclusion

The objective of this project was to provide insight into the field of over-parameterized neural networks and lazy training. Our research has extended our understanding of these complex systems by empirically studying the behavior of neural networks with increasing width, and defining a weight-movement significance level that characterizes the onset of lazy training. Additionally, we have explored the applicability of lazy training in deep neural networks of different architectures, offering critical insights into the behavior of such networks past what we have seen in the literature.

Our experimental methodology involved gradually increasing the width of the neural networks and monitoring the degree of weight movement during the training process. Through this, we were able to establish a weight-movement significance level, which provides a practical metric for characterizing the onset of lazy training. Furthermore, our analysis of deeper neural networks provided important insights into the applicability of lazy training beyond one hidden layer. We found that lazy training does not apply as strongly in networks with two layers of data that are not linearly separable. In addition, we made some interesting observations about the convergence rates of different layers in a multi-layer neural network, while assuming the roles that each individual layer plays as the number of weights increase. We concluded and verified that lazy training does still hold with structured data, and finally, we demonstrated compelling phenomena relating to sudden and rapid loss movement in varying-width neural networks.

The significance of our research lies in the fact that it highlights and confirms the crucial role played by network architecture in the phenomenon of lazy training. Our findings suggest that the degree of over-parameterization, combined with the number of hidden layers in the network, plays a key role in determining whether a network can effectively exhibit lazy training. Overall, our research has contributed to the body of knowledge on over-parameterized neural networks and lazy training.

All code used for this project is available on our **GitHub**.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012.

[2] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[3] D. Soudry and Y. Carmon, "No bad local minima: Data independent training error guarantees for multilayer neural networks," 2016.

[4] I. Safran and O. Shamir, "Depth-width tradeoffs in approximating natural functions with neural networks," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70, 2017, pp. 2979–2987.

[5] S. Arora, N. Cohen, and E. Hazan, "On the optimization of deep networks: Implicit acceleration by overparameterization," 2018.

[6] B. D. Haeffele and R. Vidal, "Global optimality in tensor factorization, deep learning, and beyond," 2015.

[7] Q. Nguyen and M. Hein, "The loss surface of deep and wide neural networks," 2017.

[8] Vol. 5, 2021. [Online]. Available: https://doi.org/10.1109%2Ftetci.2021.3100641

[9] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, pp. 303–314, 12 1989.

[10] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[11] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Transactions on Information Theory*, vol. 39, no. 3, 1993.

[12] S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai, "Gradient descent finds global minima of deep neural networks," 2019.

[13] Y. Li and Y. Liang, "Learning overparameterized neural networks via stochastic gradient descent on structured data," 2019.

[14] S. S. Du, X. Zhai, B. Poczos, and A. Singh, "Gradient descent provably optimizes over-parameterized neural networks," 2019.

[15] L. Chizat, E. Oyallon, and F. Bach, "On lazy training in differentiable programming," 2020.

[16] D. Zou, Y. Cao, D. Zhou, and Q. Gu, "Stochastic gradient descent optimizes over-parameterized deep relu networks," 2018.

[17] S. Arora, S. S. Du, W. Hu, Z. Li, R. Salakhutdinov, and R. Wang, "On exact computation with an infinitely wide neural net," 2019.

[18] Y. Cao and Q. Gu, "Generalization error bounds of gradient descent for learning over-parameterized deep relu networks," 2019.

[19] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018.

[20] Y. Cho and L. Saul, "Kernel methods for deep learning," in *Advances in Neural Information Processing Systems*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, Eds., vol. 22, 2009.

[21] Z. Allen-Zhu, Y. Li, and Z. Song, "A convergence theory for deep learning via over-parameterization," 2019.

[22] M. Zhou, R. Ge, and C. Jin, "A local convergence theory for mildly over-parameterized two-layer neural network," 2021.

[23] Y. Bengio and Y. Lecun, *Scaling learning algorithms towards AI*. MIT Press, 2007.

[24] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[25] Y. Bengio, *Learning Deep Architectures for AI*. Now Publishers Inc., 2009, vol. 2, no. 1.

[26] M. Telgarsky, "Representation benefits of deep feedforward networks," *arXiv preprint*, 2015.

[27] R. Eldan and O. Shamir, "The power of depth for feedforward neural networks," 2016.