Quo Vadis, Sharpness? Understanding Progressive Sharpening and Edge of Stability

Raphaella Diniz rcd3@sfu.ca
Ryan Wu rwa106@sfu.ca
Zhuo Ning zna3@sfu.ca
Sai Raj Kishore Perla srp7@sfu.ca

Abstract

Progressive Sharpening and *Edge of Stability* are two recently discovered regimes [Cohen et al., 2021] which govern the training dynamics in neural networks. However, their behavior is inconsistent with several widespread presumptions in the field of optimization, and as a result very little is know about them. Motivated by their importance and a lack of comprehensive understanding of these regimes, in this work, we conduct an empirical study to further the understanding of these regimes. Specifically, we investigate the reasoning behind *Progressive Sharpening*, the effect of various neural network parameters on these regimes, and the effect of these regimes on the behaviour of neural networks in a full-batch gradient descent setting. We hope, our findings will help reason better about *Progressive Sharpening* and the observed training dynamics at the *Edge of Stability*.

1 Introduction

Traditional analyses of gradient descent in optimization theory demonstrate how the optimal value of the learning rate (η) for training a neural network (NN) is related to the largest eigenvalue of the training loss Hessian, also known as the sharpness, S. This relationship, as shown in Section 2, can be demonstrated with the inequality: $S < 2/\eta$. Moreover, the training is "stable" and the training loss is expected to decrease monotonically when the inequality is satisfied, but quickly diverge if the sharpness goes beyond $2/\eta$.

Contrary to this belief, in a recent work by [Cohen et al., 2021] the authors show that this hypothesis, surprisingly, does not hold true for neural network training. Particularly, they observed two important phenomena. The first, dubbed *Progressive Sharpening*, is when the sharpness steadily increases throughout training until it reaches the instability cutoff $2/\eta$. The second, dubbed *Edge of Stability (EoS)*, is that the sharpness hovers at $2/\eta$ for the remainder of training while the training loss continues decreasing, albeit non-monotonically.

While [Cohen et al., 2021] conducted a range of experiments to support their claims, we believe that further experiments and reasoning on this topic can advance the understanding behind the cause and effects of these regimes. As neural network applications continue to differ in model architecture, loss functions, and optimization techniques, it is more important than ever to gain a comprehensive understanding of *Progressive Sharpening* and the dynamics at *Edge of Stability*, the effect of various neural network parameters on these regimes, and the effect of these regimes on the behaviour of the neural networks.

In this work, we conduct an extensive empirical study encompassing all potential aspects which we believe might have an impact on or get impacted by *Progressive Sharpening* or the dynamics at the *Edge of Stability*. Our work focuses on:

1. ***Reasoning Progressive Sharpness***: We develop an hypothesis behind *Progressive Sharpening* and validate it through empirical results.

2. ***Dataset complexity***: We analyze the relation between dataset complexity and sharpness by varying various data attributes such as the number of training samples, class imbalance, etc.
3. ***Loss function***: We analyze the effects of different loss functions on the sharpness and its dynamics at the *Edge of Stability*.
4. ***Model attributes***: We investigate the effect of model architecture and the potential influence of under/over-parameterization on the behavior of instability.
5. ***Generalization capacity***: We clarify the relationship between sharpness and the ability of the model to generalize to new unseen data.
6. ***Instability over each weight***: To better understand how the model is affected at the *Edge of Stability*, we investigate the evolution of each weight during the training.

2 Background

In this section, we review the stability properties of gradient descent on quadratic functions. Later, we will see how these properties relate to the stability of gradient descent on neural training objectives.

Let us first analyze a simple one-dimensional quadratic, $f(x) = \frac{1}{2}ax^2 + bx + c$. This function has a optimum at $x^* = \frac{-b}{a}$. Consider running gradient descent with step size η starting from x_0. The update rule is $x_{t+1} = x_t - \eta(ax_t + b)$, which means that the error $x_t - x^*$ evolves as $(x_{t+1} - x^*) = (1 - \eta a)(x_t - x^*)$. Therefore, the error at step t is $(x_t - x^*) = (1 - \eta a)^t(x_0 - x^*)$, and so the iterate at step t is $x_t = (1 - \eta a)^t(x_0 - x^*) + x^*$. If $a > \frac{2}{\eta}$ or $a < 0$, then $\|1 - \eta a\| > 1$, so the sequence $\{x_t\}$ will oscillate around x^* with exponentially increasing magnitude, and diverge.

Now, consider a general d-dimensional case, where $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c$. Let $(a_i, \mathbf{q}_i)$ be the i-th largest eigenvalue/eigenvector of $\mathbf{A}$. As shown in Appendix A, when the gradient descent iterates $\{\mathbf{x}_t\}$ are expressed in the special coordinate system whose axes are the eigenvectors of $\mathbf{A}$, each coordinate evolves separately. In, particular, the coordinate for each eigenvector $\mathbf{q}_i$ namely $\langle \mathbf{q}_i, \mathbf{x}_t \rangle$, evolves according to the dynamics of gradient descent on a one-dimensional quadratic objective with a second derivative a_i. Therefore, if $a_i > \frac{2}{\eta}$, then the sequence $\{\langle \mathbf{q}_i, \mathbf{x}_t \rangle\}$ will oscillate with exponentially increasing magnitude; in this case, we say that the iterates $\{\mathbf{x}_t\}$ diverge along the direction $\mathbf{q}_i$.

Polyak momentum [Polyak, 1964] and Nesterov Momentum [Nesterov, 1983, Sutskever et al., 2013] are notable variants of gradient descent which often improve the convergence speed. On quadratic functions, these two algorithms also diverge if the sharpness exceeds a certain threshold, which we call the "maximum stable sharpness", or MSS. In particular, we show in Appendix A that gradient descent with step size η and momentum parameter β diverges if the sharpness exceeds:

$$\text{MSS}_{Polyak}(\eta, \beta) = \frac{1}{\eta}(2 + 2\beta); \qquad \text{MSS}_{Nesterov}(\eta, \beta) = \frac{1}{\eta}\left(\frac{2 + 2\beta}{1 + 2\beta}\right)$$

3 Related Work

Instability in the loss while training deep neural networks was observed way before [Cohen et al., 2021], in [Wu et al., 2018] and [Xing et al., 2018]. These works considered both stochastic gradient descent (SGD) and full-batch gradient descent (GD). During regular full-batch GD training, the authors observed that the loss decreases monotonically until it reaches a certain point and then becomes unstable, decreasing in a non-monotonic fashion. Furthermore, Wu et al. [2018] also suggested that this behavior is related to the sharpness of the loss function.

Later papers offered further insight into the Edge of Stability (EoS) phenomenon. In [Ma et al., 2022], the authors explained EoS theoretically using the subquadratic growth of loss functions. In [Damian et al., 2022], the authors proposed a new concept called "self-stabilization" to explain the dynamics at EoS. They used the cubic Taylor approximation of the training loss to show that as the iterates in the training process diverge in the direction of the top eigenvector of the Hessian matrix, the cubic term in the local Taylor expansion of the loss function causes the curvature to decrease, restoring stability to the training process.

Subsequent research further explored the dynamics at EoS in adaptive gradient methods. In [Cohen et al., 2022], the authors studied the dynamics at EoS in "frozen Adam", a version of the Adam optimization algorithm in which the preconditioner is frozen at its current value. They found that the regime of "Adaptive Edge of Stability" (AEoS) differs from that of non-adaptive methods and that adaptive methods are able to enter high-curvature regions of the loss landscape that are inaccessible to non-adaptive methods. The authors also empirically demonstrated the characteristics of AEoS in a mini-batch Adam setting.

While there have been some significant works on understanding the dynamics at the *Edge of Stability* since [Cohen et al., 2021], the reasoning behind *Progressive Sharpening* still remains an open question.

4 Understanding *Progressive Sharpness* and *Edge of Stability*

4.1 Reasoning Progressive Sharpness

In this section, we develop an hypothesis for Progressive Sharpening and validate it through an empirical study. To do so, we ask a series of questions, whose answers, backed by experimental results, contribute towards the hypothesis.

We start with, *What is Sharpness?* Sharpness is the maximum eigenvalue of the training loss Hessian. *What is the Hessian?* The Hessian is the Jacobian of the gradient. But more importantly, it is a function of the input data, network weights, loss function, and model architecture. Now, *Which of these attributes actually change during gradient descent training?* Only the network weight values, everything else remains fixed throughout training. As a result, we can say that the Sharpness is directly correlated to the network weights. However, the network weights at any iteration t are dependent upon of the weight initialization, optimizer, learning rate, loss function, model architecture, the iteration number t, etc. To state our hypothesis formally:

Hypothesis: *The Sharpness, S, at any iteration t is positively correlated to the current values of the network weights in a full batch gradient descent setting.*

Before we dive into the experimental results, let us answer one last question. According to our hypothesis, sharpness is positively correlated to the network weights. *So, is progressive sharpening bad? Is increasing weights bad?* Not necessarily. Consider a simple function $f(x) = (x-4)^2$ which is minimized at $x^* = 4$. Now, if x_0 is initialized to a very small value, say 0, the value of x will increase during training to reach x^*. However, if x_0 is initialized to a very large value, say 10, the value of x will decrease during training to reach x^*.

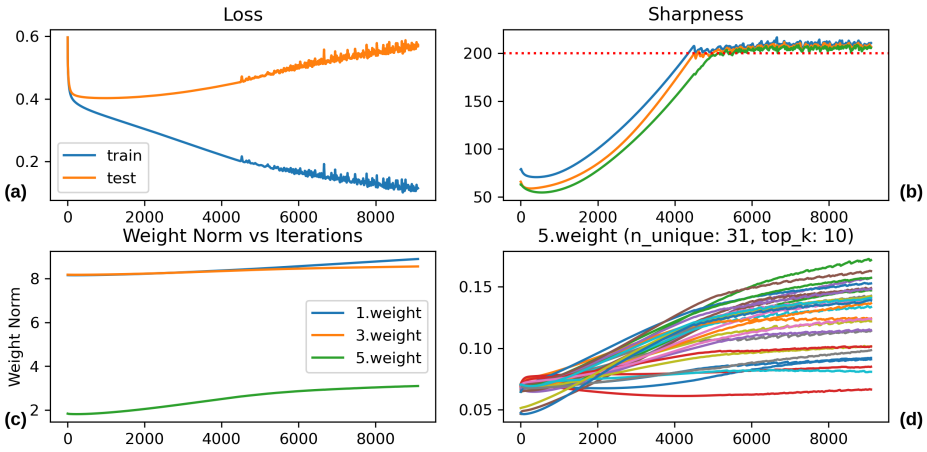

Figure 1: Results using the default `Config_1` configuration. **(a)** The training and testing loss **(b)** The evolution of the top 3 eigenvalues of the training loss Hessian. The blue curve represents the sharpness, S **(c)** The L_2-norm of the layer weights **(d)** The evolution of individual weights which were in the top-10 highest values at least once during the entire training.

We use `Config_1` (see Appendix B) for these set of experiments, and a result with the default configuration is given in Figure 1. To validate our hypothesis, we first plot the evolution of the L_2-norm of the layer weights and observe an close correlation between the evolution of the weights from the last layer and the Sharpness. We then plot the evolution of individual weights values from the last layer, which were in the top-10 highest values at least once during the entire training. We observe two interesting things from these plots. First, the overall trajectory of the evolution of these weights closely follows the trajectory of the evolution of sharpness (the overall shape and not the exact values). Second, as the sharpness crosses the stability threshold, just like the loss, most of the weights start oscillating. While the weight values still continue to grow/decay towards their optimum value, they now do so in a *"non-smooth"* fashion.

To strengthen our validation, we run a wide variety of experiments and our observations are summarized below:

- *Varying Initializations* (Appendix C): We investigate the progressive sharpness on different weight initializations. We observe that the sharpness increases significantly in scenarios where the weights are initialized at a value below their (global or local) optimal value. In scenarios where the weights are initialized with higher values than their optimal and have to decrease during the training, we observe that the sharpness is low and does not reach the $2/\eta$ threshold.
- *Network Width* (Appendix D): We vary the width of the neural network to analyze the impact on the sharpness. Even by changing the network width, we find that the model weights follow the sharpness tendency.
- *Momentum* (Appendix E): We present experiments using Nesterov and Polyak momentum. For the Nesterov momentum, the sharpness reaches the instability threshold much sooner than on the Polyak momentum. As a result, the loss is unstable for most of the training. The weight values are also unstable, despite still converging to some target value. When applying Polyak momentum, the sharpness reaches the threshold only by the end of the training, resulting in a monotonically decreasing loss along almost all iterations.
- *Regularization* (Appendix F): In these experiments, the model is trained with L_1 and L_2 regularizations. While the sharpness increases until reaching the instability threshold, on the setting with L_2-norm it does not happen, while it happens in the L_1-norm setting. Once the L_2-norm penalizes more severely bigger weights, we hypothesize that larger weight values play an important role on the behavior of Progressive Sharpening and the Edge of Stability.
- *Linear and Logistic Regression* (Appendix G): Simpler models, such as Linear Regression and Logistic Regression, were considered when analyzing the sharpness evolution. We found that the L_2-norm of the layer weights presents distinct behavior in each model. While the norm decreases on the Linear Regression model, it increases continually in Logistic Regression.

See Appendix H for our experiments on neural network divergence and Appendix I for our intuitive explanation behind the dynamics at the *Edge of Stability*.

4.2 The Effect of Dataset Complexity

In this next section, we will investigate what effect the complexity of the dataset has on EOS behavior. In previous works related to EOS, we have not seen studies related to dataset complexity, but we have reasons to believe that EOS behavior is also correlated with datasets. As pointed on Cohen et al. [2021], the EoS behavior is observed on less intensity on simpler datasets. During practical NN training, is is not unusual to handle dataset imbalance or noisy data, for example. So we feel it is necessary to study the impact of these behaviors on EOS. The full figures for these experiments are located at Appendix J.

- *Number of samples*: The experiment is designed to determine whether a smaller dataset volume has any effect on EoS, as measured by training and testing loss, training and testing accuracy, and sharpness values. To conduct the experiment, we reduce the volume of the dataset from 10% to 90% and compare the results with the original dataset while keeping the same proportion of classes. The results show that a smaller dataset volume does indeed decrease the training time, but has no effect on EoS.

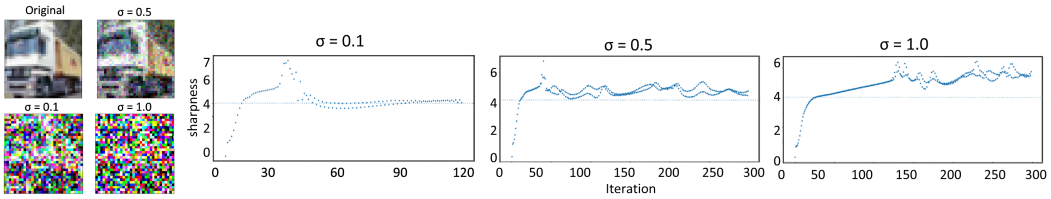

Figure 2: Sharpness for different noise augmentations in training NN.

- *Input dimension*: This experiment is designed to determine whether a lower-dimensional dataset is more stable than a higher-dimensional dataset. To conduct the experiment, we reduce the size of the data from 10% to 90% and compare the results with the original dataset. The results show that a dataset with a lower input dimension tends to be more stable than a higher-dimension dataset.
- *Class Imbalance*: This experiment is designed to determine whether a dataset with a highly imbalanced distribution of classes has any effect on EoS. To conduct the experiment, we increase the proportion of one class and reduce the proportions of the other classes. The results show that as the data becomes more imbalanced, EoS reaches the blowup phase more quickly. However, the changes are not significant and the reasons for this behavior are not clear.
- *Noise*: This experiment is designed to determine whether noise has any effect on EoS, and if so, how much noise is necessary to cause an effect. To conduct the experiment, we add Gaussian noise with different standard deviations to the data. The results show that as the amount of noise increases, EoS also increases (see Figure 2). This is thought to be because adding noise makes the underlying truth function more complex and, as noted by Cohen et al. [2021], complex datasets and models experience stronger instability during the EoS.

4.3 The Effect of Loss Functions.

The choice of loss function appears to dictate the behavior of sharpness after reaching the edge of stability. We tested several loss functions and analyzed their affect on the regimes. In particular, cross-entropy displays growing highly erratic spikes in sharpness followed by a return to EoS. However, in the long term and with small step sizes, Cross-Entropy actually displays a tendency to approach the EoS and dip below towards the end of training. Tests are performed on the `Config_1` configuration, with different loss functions. Full figures of train and test loss and accuracy are in Appendix K.

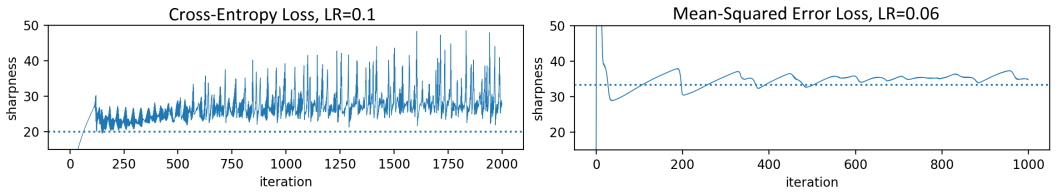

Figure 3: Sharpness for Cross Entropy and Mean Squared Error. $\eta = 0.1$ and $\eta = 0.06$. Note: For cross entropy, η is large enough that the decreasing behavior does not visibly occur yet.

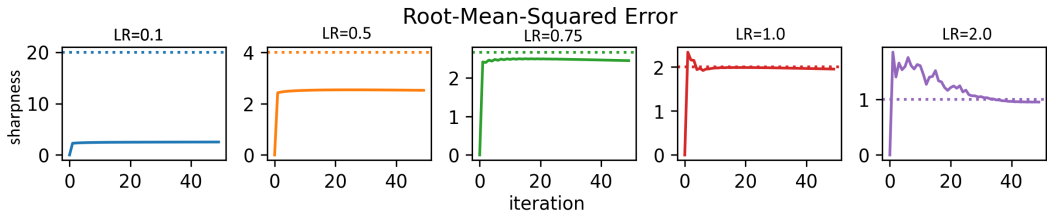

Figure 4: Root-Mean Squared Error

- *Cross-Entropy/Logistic Loss*. The magnitude of oscillations increase but sharpness does not diverge completely. This results in noisier sharpness behavior, affecting train and test loss (Figure 3). The cross-entropy sharpness hover above the $2/\eta$ threshold. However, with very small step sizes that prevent sharpness from quickly reaching the EoS, sharpness actually decreases towards the end of training. Soudry et al. [2018] and Cohen et al. [2021] discusses this phenomenon in further detail, examining how decreasing logistic loss leads to gradient descent attempting to drive the cross entropy loss to zero. The logistic loss and the second derivative of the logistic loss decreases once enough examples are classified correctly and the training objective optimizes. This means that although progressive sharpening still occurs and the weights continue to increase, the leading eigenvalue of the Hessian is pulled down and exhibits the tendency to drop near the end of training.

- *Mean-squared Error*. Unlike Cross-Entropy, Mean-Squared Error loss does not demonstrate a decrease in sharpness near the end of training (Figure 3). Mean-Squared Error exhibits more stable behavior, where the magnitude of oscillations in sharpness after reaching the EoS decreases or does not grow over time. A smaller distance between sharpness and EoS is also observed. It also begins to diverge earlier, at η 0.1 in our experiments.

- *Root-Mean-squared Error*. RMSE behaves similarly to mean-squared error, but sharpness does not initially blow up and quickly converge to the $2/\eta$ threshold unless step size is large enough ($\eta = \ 0.8$ for Figure 4). Because the loss is squared, this allows larger LR to be used, demonstrated by the rightmost example which shows $\eta = 2$ for this set of parameters whereas MSE with the same parameters diverges with a step size of 0.1.

4.4 The Effect of Model Attributes.

Cohen et al. [2021] analyze the sharpness of deep neural models and its relation to the depth of the model. The authors report that the sharpness was not observed in shallow models. Following their findings, we also did not observe the edge of stability on a shallow model such as logistic regression. It was necessary at least one hidden layer to observe such behavior. Based on this observation, we investigate if the rise of sharpness and edge of stability are related to the over-parametrization of the model. To increase the number of weights, we changed the lenght of the model. Also, we considered models with and without bias values. We did not perform experiments changing the width of the model, since Cohen et al. [2021] already presented this experiment.

Changing the number of parameters on the model directly changes the loss function, so the sharpness is expected to change. Indeed, we observed changes in the loss function during the training. More specifically, even for a simple database such as MNIST, we noted that the instability is distinct between different networks, with the increased loss occurring on varying patterns and intensities. However, we did not note any direct relationship between the number of parameters and the instability of the loss during the edge of stability or the rate with the sharpness increases. The results are presented in Appendix L.

4.5 Generalization Capacity.

In this set of experiments, we aim to explore a possible relationship between instability and the generalization capacity of the model. Although [Cohen et al., 2021] present graphs regarding the evolution of the training losses, no analysis was done on the test sets and all the important discussion is centered around the training dynamics. We investigate the test accuracy and loss during the training to see if there is a correlation with the instability of training.

We also tracked of the performance of the model on a different but related dataset. For this set of experiments, we trained the model on one dataset and used a different, but similar, dataset to compute the test loss and accuracy. We conducted the experiments in a domain adaptation setting, where labeled data from a source (training) dataset and unlabeled data from a target (test) dataset are available. The goal is to predict the label for the target data accurately. We applied the domain adaptation method DRCN Ghifary et al. [2016] to perform the alignment between features. DRCN trains an autoencoder to reconstruct the data from the target domain jointly with a classifier to predict the label from the source domain. After DAN was trained, we extracted from the autoencoder module the latent representation of both datasets.

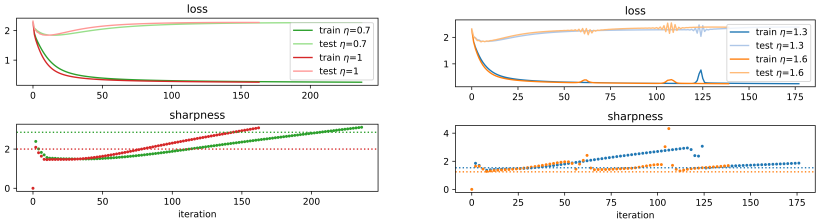

Figure 5: Loss (upper) and sharpness (bottom) on different learning rates. We did not observe a significant change in accuracy after reaching the $2/\eta$ threshold. The training and test loss present instability during the same periods. Experiments run using `Config_2` setting presented on Appendix B

The experiments were performed considering the datasets MNIST and USPS. First, we compared if the test accuracy and loss has a significant impact immediately after the model reaches the edge of stability. No noticeable alteration was noted in the test results, meaning that there is possibly no difference between results before and after the model exceeds the edge of stability (see examples on Figure 5).

We notice that the instability in the test data follows the same pattern as the training. In other words, every time there is a sudden increase in the training loss, the test loss also present instability, regardless of the training time. This result is particularly useful since it is possible to know in advance the performance trend of the model on unseen data, especially on domain adaptation methods where the labeled data is not available to assess the model's performance. Note that the instability does not indicate the good or bad performance of the model on test data, only its momentary instability (see right column on Figure 5).

4.6 Effect of Instability Over Each Weight.

To better understand the causes and impacts of the edge of stability, we investigated the evolution of each weight during the training. We aim to verify if the instability causes (or is caused by) a sudden change in the values of some groups of weights or if it affects the whole model.

Considering MNIST, which is a simple dataset that can be trained in a few epochs, it is possible to note that the instability happens in "waves". The loss is affected for a few epochs, then decreases monotonically for a few epochs. We followed the increase or decrease tendency of each weight during the training. Then, during periods of instability, where the loss is irregular, we checked if the trend of each weight is kept or if it also behaves erratically, increasing and decreasing (see examples on the right column of Figure 5.

During periods of instability, more than 95% of the weights are affected. It means, for example, that a weight with an increase trending sees its value decrease and increase sequentially until the end of the instability, where it continues to rise again. The magnitude of the change is proportional to the weight's value. With this result, we conclude that the instability affects (or is affected by) almost all the model.

MNIST is known for being an easy dataset that can be trained with a simple regression model. Adopting an over-parameterized model, such as a fully connected neural network with two hidden layers, we expect that some weights have much more impact on the loss. The discovery that most weights face instability raises the suspicion that the edge of instability may have no relationship with the importance of each weight.

5 Conclusion and Future Work

In this work, we aimed to conduct a more diverse set of experiments that were not covered or were covered superficially in [Cohen et al., 2021]. We hope our results bring new insights into the reason behind *Progressive Sharpening* and the dynamics at the *Edge of Stability*.

5.1 Summary of Findings

1. ***Reasoning Progressive Sharpness*:** We discussed the relationship between the sharpness and model weights and how weight values are affected by model attributes such as the learning rate, loss function, model architecture, etc. We developed an hypothesis which postulates that changes in weights correlates to changes in sharpness. Our experiments informed our observations on tendency of the weights to follow the trajectory of the sharpness, a behavior which persists across various experiments and differing parameters. As the sharpness begins oscillating after reaching the *Edge of Stability*, we also note subsequent oscillations in the weights as well.
2. ***Dataset complexity*:** We analyse the relationship between the dataset complexity and sharpness. Our findings demonstrate how changes in input dimension increases stability due to fewer weights. Noise causes sharpness to oscillate at a threshold above the *Edge of Stability*, and as the data becomes more imbalanced, EoS reaches the blowup phase more quickly.
3. ***Loss function*:** We observed how the choice of loss function affects behavior after reaching the *Edge of Stability*, and the respective variations and attributes of different loss functions. With small step sizes, We also discuss how progressive sharpening still occurs and the weights continue to increase, but the leading eigenvalue of the Hessian is pulled down and exhibits the tendency to drop near the end of training.
4. ***Model Attributes*:** Modifying model parameters changes the behavior of instability, although a direct pattern or relationship as not observed for either.
5. ***Generalization Capacity*:** We investigate the test accuracy and loss during the training to see if there is a correlation with the instability of training. An autoencoder was trained on datasets MNIST and USPS,and we discussed the lack of difference between results before and after exceeding EoS. Also, increases in the training loss are reflected in the test loss, regardless of the training time, suggesting a possibility for prediction of trends.
6. ***Analysis of Instability over each weight*:** During spikes in stability, we observed more than 95% of the weights individually have subsequent spikes as well. This means the weight values change sequentially until the end of the instability, where it continues to rise again. The magnitude of the change is proportional to the weight's value. With this result, we conclude that the instability affects (or is affected by) almost all of the model.

5.2 Challenges and Future Work

1. *Theoretical reasoning behind the correlation between weight and sharpness*. Demonstrating or validating a mathematical intuition for the relationship would present more insights into the relationship between weights and sharpness.
2. *Consequences for GD optimization choices and practices for Neural Networks*. Future work could focus on step size heuristics to avoid the EoS, or analysis of adaptive step sizes.
3. *Cross-Entropy Loss*. Further analysis into the behavior of cross entropy loss and reasoning for decreasing sharpness near the end of training could be connected to our findings and optimization techniques.
4. *Datasets and Architectures*. Although we performed experiments on simple visual datasets (USPS, MNIST, and CIFAR10k), we decided not to test larger datasets due to a limited time-frame and computational resources. Experiments on different neural network architectures would also be a good topic for future analysis.
5. *Hessian Calculation*. As the size of a model grows, calculating the Hessian quickly becomes an exercise in futility. A simple neural network to classify MNIST has 40k weights, and the subsequent Hessian has 1.6 billion values. Approximations are still expensive and may not be very accurate as well. We attempted utilizing the PyHessian library to mitigate the computational load but observed no significant increase in speed or accuracy.
6. *Mini-Batch Gradient Descent*. Our experiments evaluate the full-batch version of gradient descent. Our findings and conclusions are not verified for smaller batch sizes.
7. *Shallow Networks and Simple Models*. Investigate why the behavior of Progressive Sharpness and Edge of Stability is less prevalent in shallow networks.

References

Jeremy M. Cohen, Simran Kaur, Yuanzhi Li, J. Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. *CoRR*, abs/2103.00065, 2021. URL `https://arxiv.org/abs/2103.00065`.

Jeremy M. Cohen, Behrooz Ghorbani, Shankar Krishnan, Naman Agarwal, Sourabh Medapati, Michal Badura, Daniel Suo, David Cardoze, Zachary Nado, George E. Dahl, and Justin Gilmer. Adaptive gradient methods at the edge of stability, 2022. URL `https://arxiv.org/abs/2207.14484`.

Alex Damian, Eshaan Nichani, and Jason D Lee. Self-stabilization: The implicit bias of gradient descent at the edge of stability. *arXiv preprint arXiv:2209.15594*, 2022.

Saber N Elaydi. An introduction to difference equations. *Springer-Verlag, New York*, 10:978–1, 1996.

Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In *European conference on computer vision*, pages 597–613. Springer, 2016.

Gabriel Goh. Why momentum really works. *Distill*, 2(4):e6, 2017.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Chao Ma, Lei Wu, and Lexing Ying. The multiscale structure of neural network loss functions: The effect on optimization and origin. *arXiv preprint arXiv:2204.11326*, 2022.

Yu E Nesterov. A method for solving the convex programming problem with convergence rate $o(\frac{1}{k^2})$. In *Dokl. Akad. Nauk SSSR,*, volume 269, pages 543–547, 1983.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19, 2018.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.

Mehmet Ali Tugay and Yalcin Tanik. Properties of the momentum lms algorithm. *Signal Processing*, 18(2):117–127, 1989.

Lei Wu, Chao Ma, et al. How sgd selects the global minima in over-parameterized learning: A dynamical stability perspective. *Advances in Neural Information Processing Systems*, 31, 2018.

Chen Xing, Devansh Arpit, Christos Tsirigotis, and Yoshua Bengio. A walk with sgd. *arXiv preprint arXiv:1802.08770*, 2018.

A Stability of Gradient Descent on quadratic functions

This appendix describes the stability properties of gradient descent (and its momentum variants) when optimizing the quadratic objective function

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c \tag{1}$$

starting from the initialization $\mathbf{x}_0$ with learning rate η.

To review, vanilla gradient descent is defined by the iteration:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta\nabla f(\mathbf{x}_t)$$

Meanwhile, gradient descent with Polyak momentum [Polyak, 1964, Sutskever et al., 2013, Goodfellow et al., 2016] is defined by the iteration:

$$\begin{aligned}\mathbf{v}_{t+1} &= \beta\mathbf{v}_t - \eta\nabla f(\mathbf{x}_t)\\ \mathbf{x}_{t+1} &= \mathbf{x}_t + \mathbf{v}_{t+1}\end{aligned}$$

where $\mathbf{v}_t$ is a "velocity" vector and $0 \le \beta \le 1$ is the momentum coefficient. For $\beta = 0$ the algorithm reduces to vanilla GD.

Finally, Nesterov momentum [Sutskever et al., 2013, Goodfellow et al., 2016] is an adaptation of Nesterov's accelerated gradient [Nesterov, 1983] for deep learning defined by the iteration:

$$\begin{aligned}\mathbf{v}_{t+1} &= \beta\mathbf{v}_t - \eta\nabla f(\mathbf{x}_t + \beta\mathbf{v}_t)\\ \mathbf{x}_{t+1} &= \mathbf{x}_t + \mathbf{v}_{t+1}\end{aligned}$$

where $\mathbf{v}_t$ is a "velocity" vector and $0 \le \beta \le 1$ is the momentum coefficient. For $\beta = 0$ the algorithm reduces to vanilla GD.

All three of these algorithms share a special property: on quadratic functions, they act independently along each Hessian eigenvector. That is, if we express the iterates in the Hessian eigenvector basis, then in this basis the coordinates evolve independent from one another under gradient descent.

Proposition 1. *Consider running vanilla gradient descent on the quadratic objective (1)* starting from $\mathbf{x}_0$. Let $(\mathbf{q}, a)$ be an eigenvector/eigenvalue pair of $\mathbf{A}$. If $a > \frac{2}{\eta}$, then the sequence $\{\mathbf{q}^T\mathbf{x}_t\}$ will diverge.

Proof. The update rule for gradient descent on this quadratic function is:

$$\begin{aligned}\mathbf{x}_{t+1} &= \mathbf{x}_t - \eta(\mathbf{A}\mathbf{x}_t + \mathbf{b})\\ &= (\mathbf{I} - \eta\mathbf{A})\mathbf{x}_t - \eta\mathbf{b}\end{aligned}$$

Therefore, the quantity $\mathbf{q}^T\mathbf{x}_t$ evolves under gradient descent as:

$$\begin{aligned}\mathbf{q}^T\mathbf{x}_{t+1} &= \mathbf{q}^T(\mathbf{I} - \eta\mathbf{A})\mathbf{x}_t - \eta\mathbf{b}\\ &= (1 - \eta a)\mathbf{q}^T\mathbf{x}_t - \eta\mathbf{q}^T\mathbf{b} \qquad (\because \mathbf{q}^T\mathbf{A} = a\mathbf{q})\end{aligned}$$

Define $\tilde{x}_t = \mathbf{q}^T\mathbf{x}_t + \frac{1}{a}\mathbf{q}^T\mathbf{b}$, and note that $\{\mathbf{q}^T\mathbf{x}_t\}$ diverges if and only if $\{\tilde{x}_t\}$ diverges.

The quantity $\tilde{x}_t$ evolves under gradient descent according to the simple rule:

$$\tilde{x}_{t+1} = (1 - \eta a)\tilde{x}_t$$

Since $\eta > 0$, if $a > \frac{2}{\eta}$ or $a < 0$ then $\|1 - \eta a\| > 1$, so the sequence $\{\tilde{x}_t\}$ will diverge.

We now, prove analogous results for Nesterov and Polyak momentum.

Theorem 1. *Consider running Nesterov momentum on the quadratic objective (1) starting from any initialization. Let* $(\mathbf{q}, a)$ *be an eigenvector/eigenvalue pair of* $\mathbf{A}$*. If* $a > \frac{1}{\eta}\left(\frac{2+2\beta}{1+2\beta}\right)$*, then the sequence* $\{\mathbf{q}^T\mathbf{x}_t\}$ *will diverge.*

Proof. the update rules for Nesterov momentum on this quadratic function are:

$$\begin{aligned}\mathbf{v}_{t+1} &= \beta(\mathbf{I} - \eta\mathbf{A})\mathbf{v}_t - \eta\mathbf{b} - \eta\mathbf{A}\mathbf{x}_t \\ \mathbf{x}_{t+1} &= \mathbf{x}_t + \mathbf{v}_{t+1}\end{aligned}$$

Using the fact that $\mathbf{v}_t = \mathbf{x}_t - \mathbf{x}_{t-1}$, we can rewrite this as a recursion in $\mathbf{x}_t$ alone:

$$\begin{aligned}\mathbf{x}_{t+1} &= \mathbf{x}_t + \beta(\mathbf{I} - \eta\mathbf{A})(\mathbf{x}_t - \mathbf{x}_{t-1}) - \eta\mathbf{b} - \eta\mathbf{A}\mathbf{x}_t \\ &= (1+\beta)(\mathbf{I} - \eta\mathbf{A})\mathbf{x}_t - \beta(\mathbf{I} - \eta\mathbf{A})\mathbf{x}_{t-1} - \eta\mathbf{b}\end{aligned}$$

Define $\tilde{x}_t = \mathbf{q}^T\mathbf{x}_t + \frac{1}{a}\mathbf{q}^T\mathbf{b}$, and note that $\mathbf{q}^T\mathbf{x}_t$ diverges if and only if $\{\tilde{x}_t\}$ diverges. It can be seen that $\tilde{x}_t$ evolves as:

$$\tilde{x}_{t+1} = (1+\beta)(1-\eta a)\tilde{x}_t - \beta(1-\eta a)\tilde{x}_{t-1}$$

This is a linear homogeneous second-order difference equation. By Theorem 2.37 in [Elaydi, 1996], since $\eta > 0$ and $\beta < 1$, if $a > \frac{1}{\eta}\left(\frac{2+2\beta}{1+2\beta}\right)$ then this recurrence diverges.

Theorem 2. *Consider running Polyak momentum on the quadratic objective 1 starting from any initialization. Let* $(\mathbf{q}, a)$ *be an eigenvector/eigenvalue pair of* $\mathbf{A}$*. If* $a > \frac{1}{\eta}(2+2\beta)$*, then the sequence* $\{\mathbf{q}^T\mathbf{x}_t\}$ *will diverge.*

Proof. Using the fact that $\mathbf{v}_t = \mathbf{x}_t - \mathbf{x}_{t-1}$, we can re-write the Polyak momentum recursion as a recursion in $\mathbf{x}$ alone:

$$\begin{aligned}\mathbf{x}_{t+1} &= \mathbf{x}_t + \beta\mathbf{v}_t - \eta\nabla f(\mathbf{x}_t) \\ &= \mathbf{x}_t + (\beta(\mathbf{x}_t - \mathbf{x}_{t-1}) - \eta\nabla f(\mathbf{x}_t)) \\ &= (1+\beta)\mathbf{x}_t - \beta\mathbf{x}_{t-1} - \eta\nabla f(\mathbf{x}_t)\end{aligned}$$

For the quadratic objective (1), this update rule amounts to:

$$\begin{aligned}\mathbf{x}_{t+1} &= (1+\beta)\mathbf{x}_t - \beta\mathbf{x}_{t-1} - \eta(\mathbf{A}\mathbf{x}_t + \mathbf{b}) \\ &= (1+\beta-\eta\mathbf{A})\mathbf{x}_t - \beta\mathbf{x}_{t-1} - \eta\mathbf{b}\end{aligned}$$

Multiplying by $\mathbf{q}^T$, we obtain:

$$\mathbf{q}^T\mathbf{x}_{t+1} = (1+\beta-\eta a)\mathbf{q}^T\mathbf{x}_t - \beta\mathbf{q}^T\mathbf{x}_{t-1} - \eta\mathbf{q}^T\mathbf{b}$$

Now, define $\tilde{x}_t = \mathbf{q}^T\mathbf{x}_t + \frac{1}{a}\mathbf{q}^T\mathbf{b}$. Note that $\mathbf{q}^T\mathbf{x}_t$ diverges if and only if $\{\tilde{x}_t\}$ diverges. It can be seen that $\tilde{x}_t$ evolves as:

$$\tilde{x}_{t+1} = (1+\beta-\eta a)\tilde{x}_t - \beta\tilde{x}_{t-1}$$

This is a linear homogeneous second-order difference equation. By Theorem 2.37 in [Elaydi, 1996], since $\eta > 0$ and $\beta < 1$, if $a > \frac{1}{\eta}(2+2\beta)$ then this recurrence diverges.

These results previously appeared in [Tugay and Tanik, 1989, Goh, 2017, Cohen et al., 2021].

B Experiment Settings

In this section we describe the training details used in our experiments. We use three base configurations for three different datasets, which are used with minor modifications throughout our experiments. The configurations are as follows:

`Config_1`:

- **Dataset**: CIFAR-10
- **Model Architecture**: 3072 ($32 \times 32 \times 3$) — 200 — 200 — 10 (Linear Layers)
- **Training Set Size**: 5,000 images
- **Test Set Size**: 10,000 images
- **Activation function**: `tanh`
- **Optimizer**: Full batch Gradient Descent
- **Momentum type**: No momentum
- **Momentum** β: —
- **Learning rate**: 0.01
- **Loss function**: Mean Squared Error (MSE)
- **Regularization**: No regularization
- **Initialization**: No initialization

`Config_2`:

- **Dataset:** MNIST
- **Model Architecture:** 784 (28×28) — x — 10 (Linear layers)
- **Training Set Size:** 6,000 images
- **Test Set Size:** 1,000 images
- **Activation function:** `tanh`
- **Optimizer:** Full batch Gradient Descent
- **Momentum type:** No momentum
- **Momentum** β**:** —
- **Learning rate:** 0.5
- **Loss function:** Cross-entropy
- **Regularization:** No regularization
- **Initialization:** No initialization

`Config_3`:

- **Dataset:** MNIST (source) and USPS (target)
- **Model Architecture:** 784 (28×28) — 5 layer — 10 (Linear layers)
- **Training Set Size:** 6,000 images
- **Test Set Size:** 2,007 images
- **Activation function:** `tanh`
- **Optimizer:** Full batch Gradient Descent
- **Momentum type:** No momentum
- **Momentum** β**:** —
- **Learning rate:** —
- **Loss function:** Cross-entropy
- **Regularization:** No regularization
- **Initialization:** No initialization

C Reasoning *Progressive Sharpening*: Initializations

In this section we experiment with different weight initialization techniques. We use `Config_1` as the default configuration with modifications as explained below.

Similar to Section 4.1, for all the figures below: **(a)** plots the training and testing loss. **(b)** plots the evolution of the top 3 eigenvalues of the training loss Hessian. The blue curve represents the sharpness, S. **(c)** plots the L_2-norm of the layer weights; and **(d)** plots the evolution of individual weights which were in the top-10 highest values at least once during the entire training.

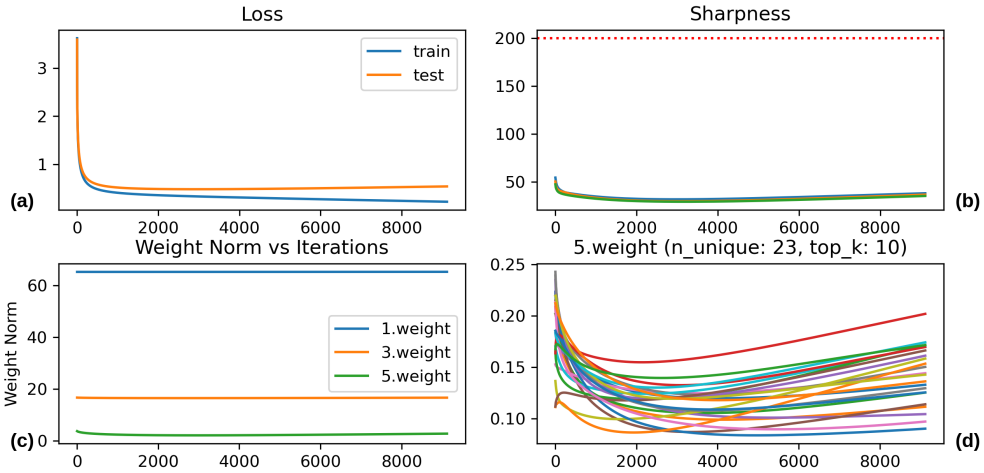

Figure 6: Weight Initialization: $\mathcal{N}(\mu = 0, \sigma = 0.0833)$. The μ and σ are chosen such that the weights are zero centered and lie between -0.25 and $+0.25$.

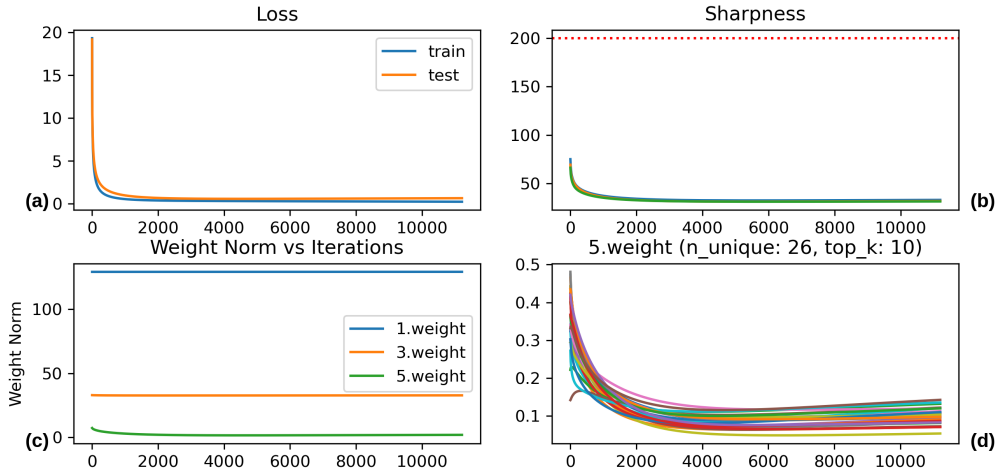

Figure 7: Weight Initialization: $\mathcal{N}(\mu = 0, \sigma = 0.165)$. The μ and σ are chosen such that the weights are zero centered and lie between -0.5 and $+0.5$.

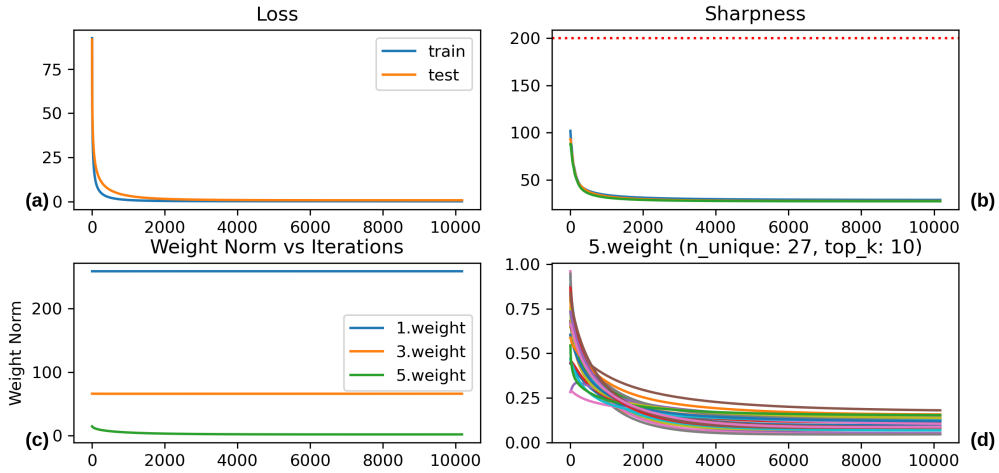

Figure 8: Weight Initialization: $\mathcal{N}(\mu = 0, \sigma = 0.33)$. The μ and σ are chosen such that the weights are zero centered and lie between -1.0 and $+1.0$.

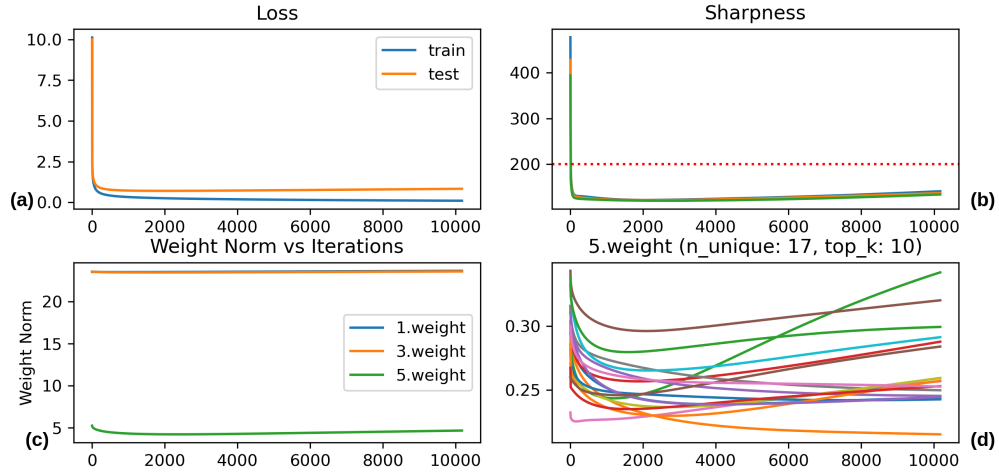

Figure 9: Weight Initialization: He Normal.

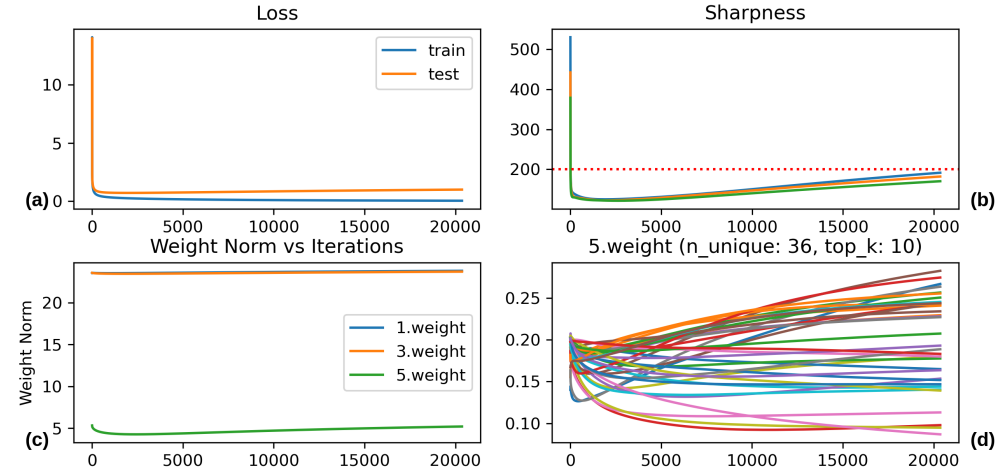

Figure 10: Weight Initialization: He Uniform.

D Reasoning *Progressive Sharpening*: Network Width

In this section we experiment with different network widths. We use `Config_1` as the default configuration and modify the network architecture to have one hidden layer with varying number of neurons.

Similar to Section 4.1, for all the figures below: **(a)** plots the training and testing loss. **(b)** plots the evolution of the top 3 eigenvalues of the training loss Hessian. The blue curve represents the sharpness, S. **(c)** plots the L_2-norm of the layer weights; and **(d)** plots the evolution of individual weights which were in the top-10 highest values at least once during the entire training.

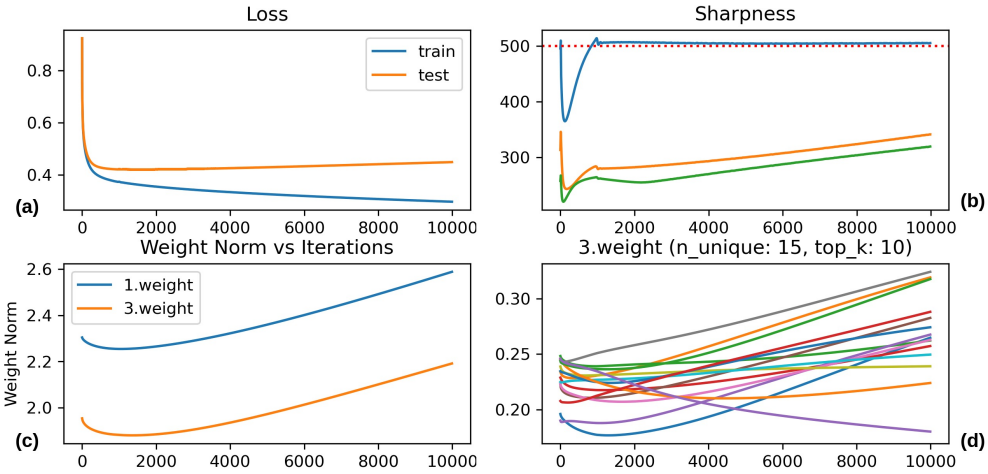

Figure 11: Network width: 16.

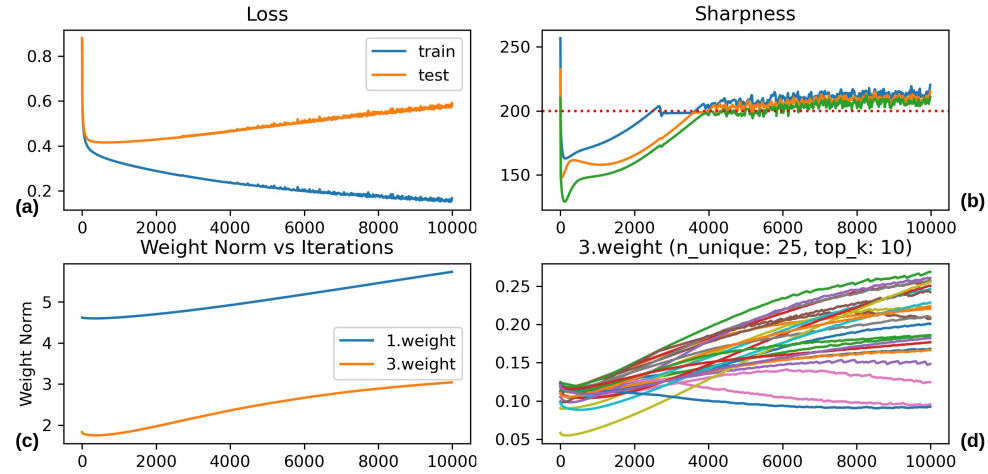

Figure 12: Network width: 64.

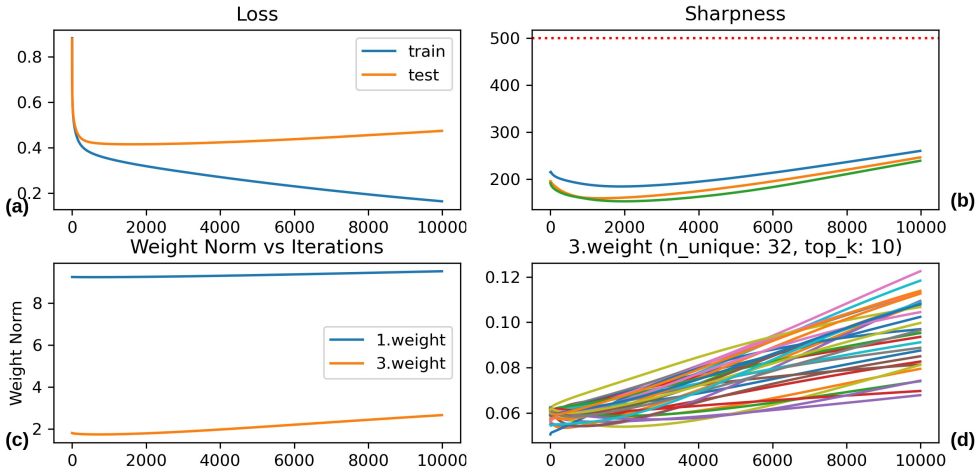

Figure 13: Network width: 256.

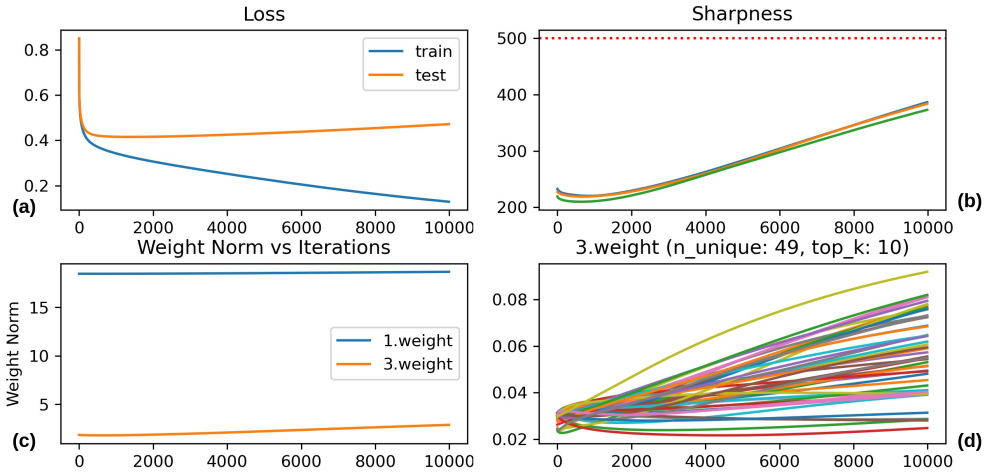

Figure 14: Network width: 1024.

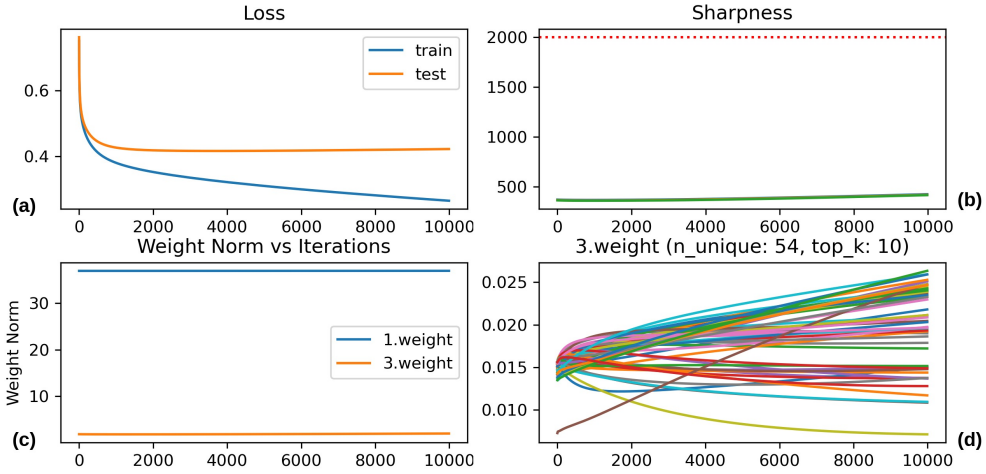

Figure 15: Network width: 4096.

E Reasoning *Progressive Sharpening*: Momentum

In this section we experiment with different momentum strategies, namely Polyak and Nesterov. We use `Config_1` as the default configuration and modify the optimizer to use momentum instead of vanilla gradient descent.

Furthermore, as discussed in Section 2, the instability thresholds for Polyak and Nesterov momentum are given by:

$$\text{MSS}_{Polyak}(\eta, \beta) = \frac{1}{\eta}(2 + 2\beta); \qquad \text{MSS}_{Nesterov}(\eta, \beta) = \frac{1}{\eta}\left(\frac{2 + 2\beta}{1 + 2\beta}\right)$$

Similar to Section 4.1, for all the figures below: **(a)** plots the training and testing loss. **(b)** plots the evolution of the top 3 eigenvalues of the training loss Hessian. The blue curve represents the sharpness, S. **(c)** plots the L_2-norm of the layer weights; and **(d)** plots the evolution of individual weights which were in the top-10 highest values at least once during the entire training.

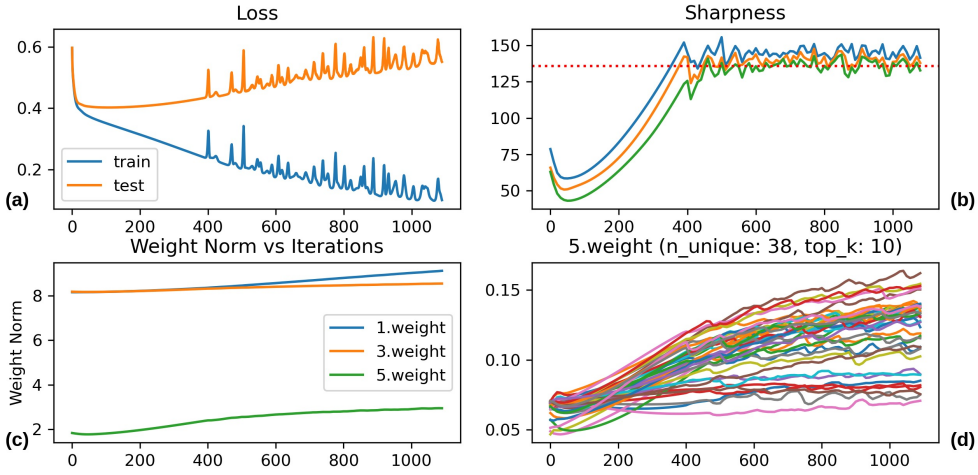

Figure 16: Using Nesterov Momentum with $\beta = 0.9$.

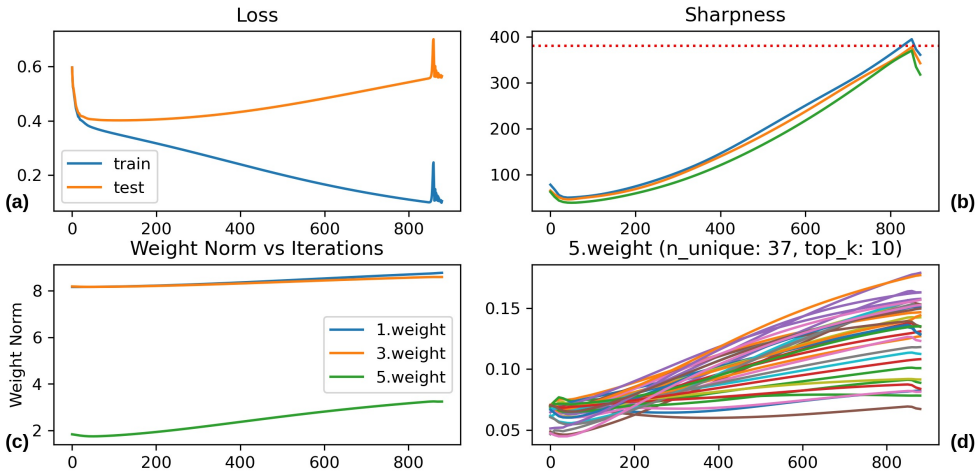

Figure 17: Using Polyak Momentum with $\beta = 0.9$.

F Reasoning *Progressive Sharpening*: Regularizations

In this section we experiment with L_1 and L_2 regularizations. We use `Config_1` as the default configuration and modify the loss function to add an additional term corresponding to the regularization.

Similar to Section 4.1, for all the figures below: **(a)** plots the training and testing loss. **(b)** plots the evolution of the top 3 eigenvalues of the training loss Hessian. The blue curve represents the sharpness, S. **(c)** plots the L_2-norm of the layer weights; and **(d)** plots the evolution of individual weights which were in the top-10 highest values at least once during the entire training.

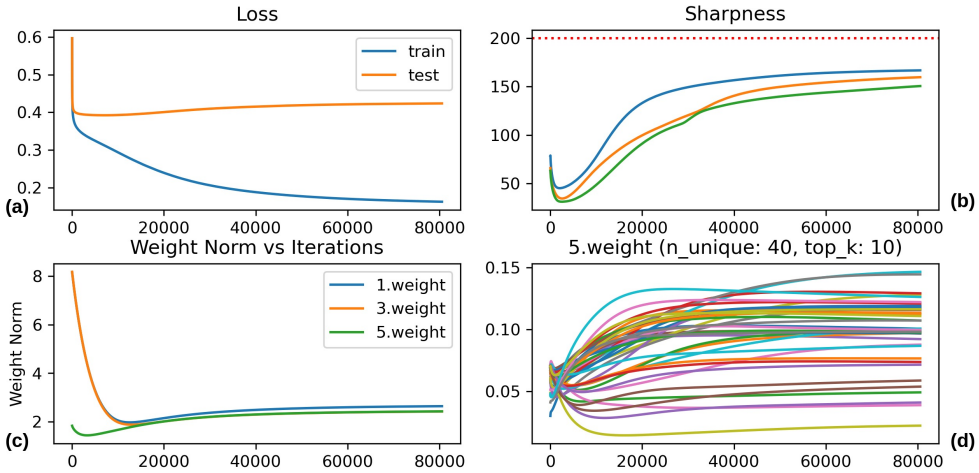

Figure 18: Using L_2 regularization with $\lambda = 10^{-2}$.

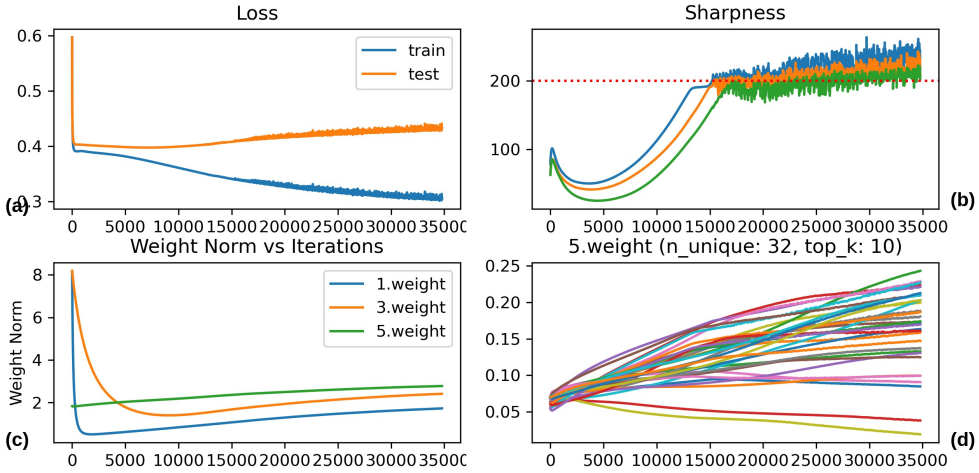

Figure 19: Using L_1 regularization with $\lambda = 10^{-6}$.

G Reasoning *Progressive Sharpening*: Simple Models

In this section we experiment with the most simple models, linear and logistic regression. We use `Config_1` as the default configuration but with a modified network with no hidden layers.

Similar to Section 4.1, for all the figures below: **(a)** plots the training and testing loss. **(b)** plots the evolution of the top 3 eigenvalues of the training loss Hessian. The blue curve represents the sharpness, S. **(c)** plots the L_2-norm of the layer weights; and **(d)** plots the evolution of individual weights which were in the top-10 highest values at least once during the entire training.

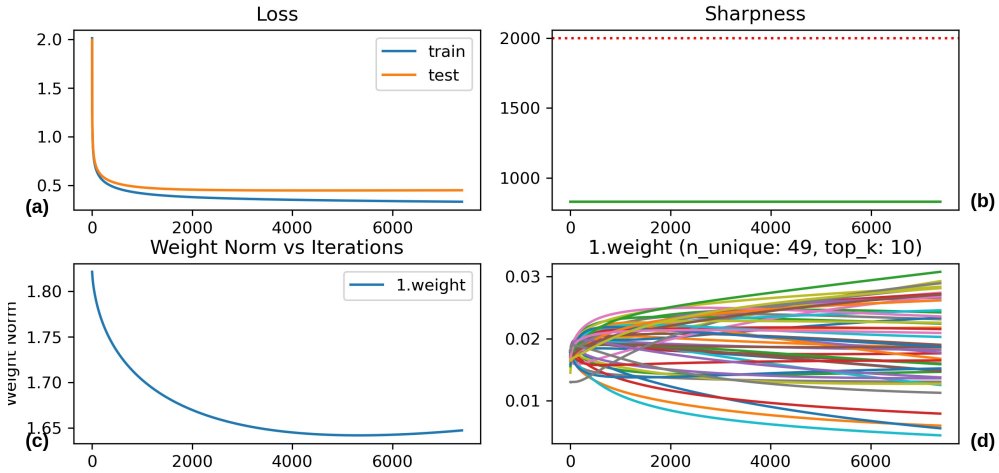

Figure 20: Linear Regression.

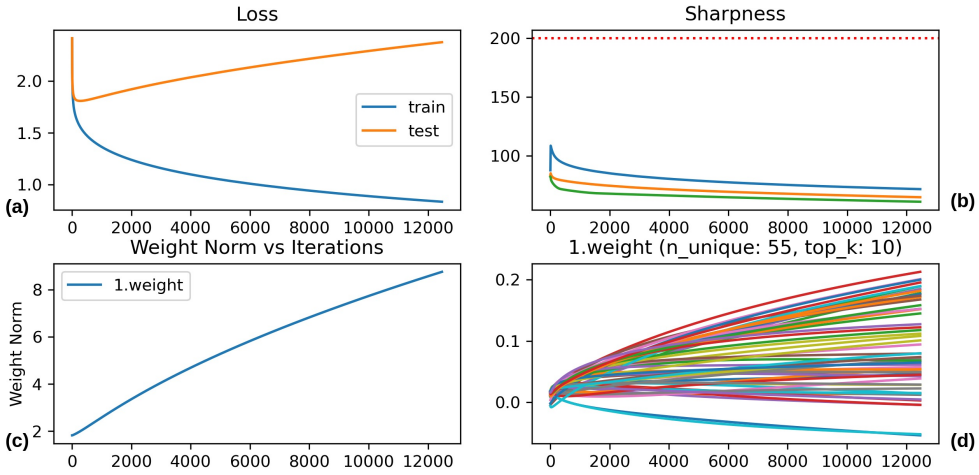

Figure 21: Logistic Regression.

H Reasoning *Progressive Sharpening*: Divergence

In this section we try to answer if the dynamics at the *Edge of Stability* prevent a neural network from divergence caused due to large learning rate or not? The answer is NO, the neural network diverges despite *Edge of Stability*. We use `Config_1` as the default configuration but with a modified learning rate, $\eta = 0.1$.

Similar to Section 4.1, for all the figures below: **(a)** plots the training and testing loss. **(b)** plots the evolution of the top 3 eigenvalues of the training loss Hessian. The blue curve represents the sharpness, S. **(c)** plots the L_2-norm of the layer weights; and **(d)** plots the evolution of individual weights which were in the top-10 highest values at least once during the entire training.

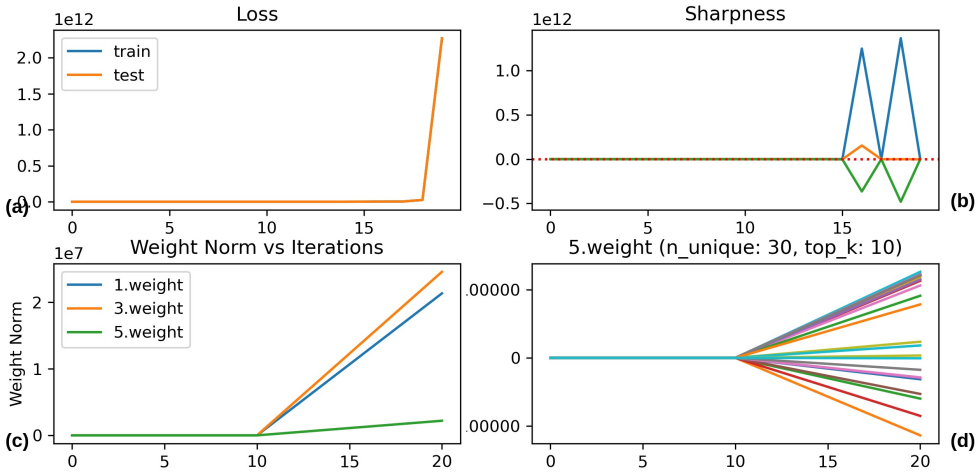

Figure 22: Neural Network divergence.

I Intuitive understanding of dynamics at *Edge of Stability*

As explained in Section 2 and Appendix A, for any n-D quadratic objective $f(\mathbf{x})$, when the gradient descent iterates are expressed in a special coordinate system whose axes are the eigenvectors of the hessian of $f(\mathbf{x})$ each coordinate evolves separately. In particular, the coordinate along each eigenvector evolves according to the dynamics of gradient descent on a one-dimensional quadratic objective.

However, neural network training objectives are not globally quadratic. While, it is common to consider the second-order Taylor approximation around any point $\mathbf{x}_0$ in the parameter space, this quadratic approximation fails to explain the neural network's behaviour at the *Edge of Stability*. Furthermore, as shown in [Damian et al., 2022], using a cubic Taylor approximation helps reason the dynamics at the *Edge of Stability*. In this section we present an intuitive explanation behind the dynamics at *Edge of Stability* and see why a quadratic approximation of the training objective fails to explain this behaviour.

Consider training a neural network $M(\theta)$, with weights θ in a full-batch gradient descent setting starting at θ^0 with a fixed learning rate η. The weight update is governed by: $\theta^{t+1} = \theta^t - \eta\nabla L(\theta^t)$, where $L(\cdot)$ is the loss function used to train the neural network. Let us analyze the evolution of any three parameters θ_1, θ_2, and θ_3 from the parameter group θ (see Figure 23). The curves in Figure 23 represent the loss with respect to each θ_i. While the loss landscapes are not necessarily parabolic, or even *strongly-convex*, in the real world, they are shown so just for illustration purposes. The initial values of θ_1, θ_2, and θ_3 are represented by "*Start*". Furthermore, since θ is randomly initialized, some θ_i's are closer to their optimum value, θ_i^*, while others are farther when initialized. For our analysis, consider θ_2 is initialized closest to θ_2^* while θ_1 is initialized the farthest from θ_1^*.

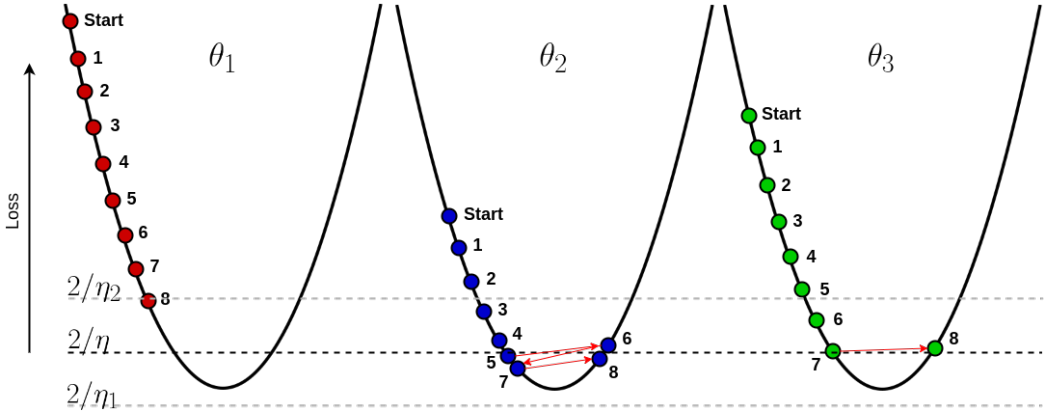

Figure 23

As we run gradient descent, the parameters get updated to move towards their optimum value every iteration. Consider the iterates till epoch 5, while all θ_1, θ_2, and θ_3 have monotonically decreased, θ_2 is closest to its optimum value followed by θ_3 and θ_1. Before we analyze the next set of iterates, it is important to understand the reason behind oscillating iterates in general. The iterates oscillate when the gradient descent update step overshoots the minima, or in other words, $|\eta\nabla_{\theta_i} L(\theta^t)| > |\theta_i^t - \theta_i^*|$.

Hypothesis. *When $|\eta\nabla_{\theta_i} L(\theta^t)| > |\theta_i^t - \theta_i^*|$ because of a large η, the iterates oscillate and diverge. However, when $|\eta\nabla_{\theta_i} L(\theta^t)| > |\theta_i^t - \theta_i^*|$ because of large gradients, $\nabla_{\theta_i} L(\theta^t)$, along one or multiple directions, the iterates oscillate but do not diverge.*

On one hand, the instability during the *Edge of Stability* indicates some level of divergence during the training. On the other hand, we observe that irrespective of the sharpness reaching the stability threshold ($2/\eta$ for gradient descent), the model still achieves a similar loss value and the weights also reach their (local or global) optimum. Unlike the example above, neural networks are not strictly-convex. Even more, *the weights are not independent*. The update of any particular weight depends on the value of other weights as well. These observations lead us to our hypothesis. If the model is locally approximated to a second-order function on a basis where the labels evolve independently, we hypothesize that some dimensions converge faster and overshoot the local/global

minima, while other dimensions still target their optimum value. This causes a momentary divergence in some dimensions. However, since the weights are not independent, the model does not diverge entirely. A diverging weight is still able to converge by considering the non-diverging weights on their update. Neural networks are many times over-parameterized. In this setting, our reported hypothesis happens dynamically, involving different parameters over time. For this reason, the instability on the loss function is observed continually and with different intensities during the edge of stability.

For experimental validation of our hypothesis see experiments in Appendix C and Appendix H.

J Influence of Data on *Edge of Stability*

Original section at 4.2.

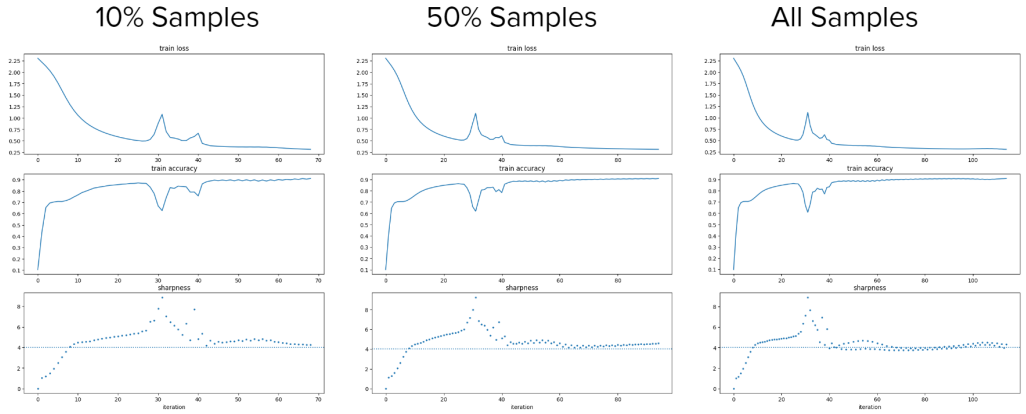

Figure 24: Loss, accuracy, and sharpness for different dataset sizes

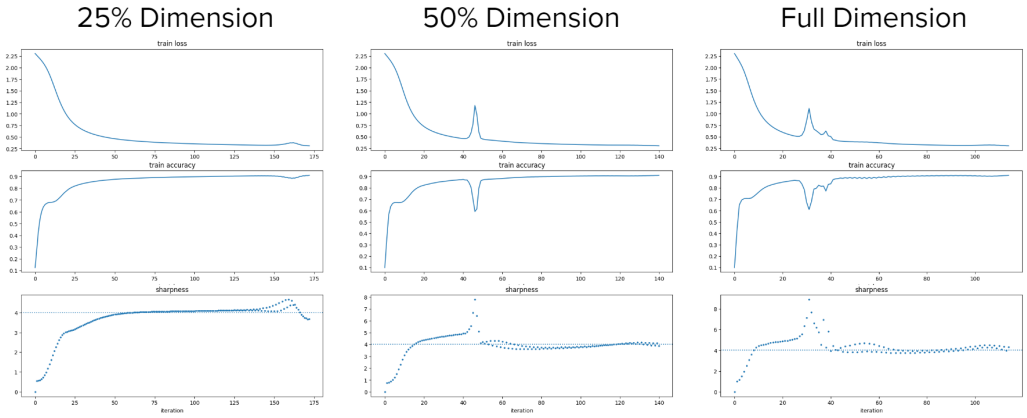

Figure 25: Loss, accuracy, and sharpness for different dimensions of input data

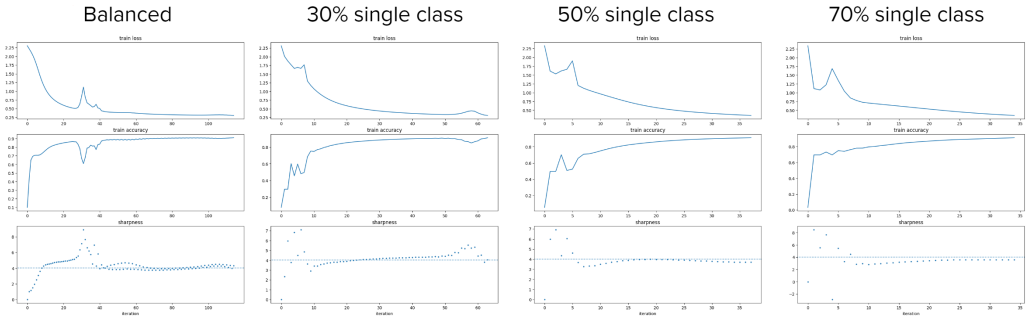

Figure 26: Loss, accuracy, and sharpness for different class imbalanced datasets

K Influencing Factors for *Edge of Stability*: Cross Entropy and Mean Squared Error

Original section at 4.3.

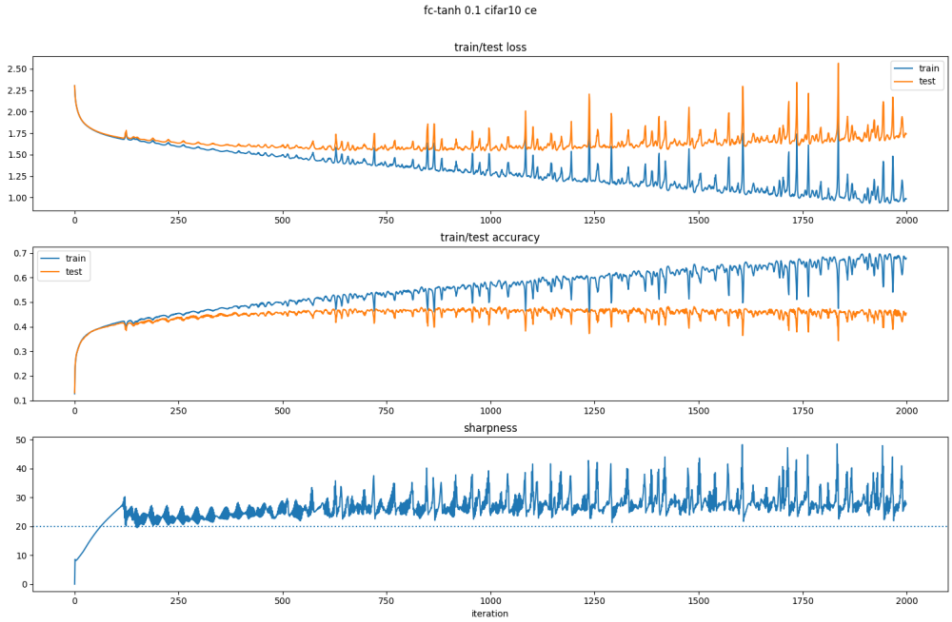

Figure 27: Test and train accuracy and loss for Cross Entropy Loss.

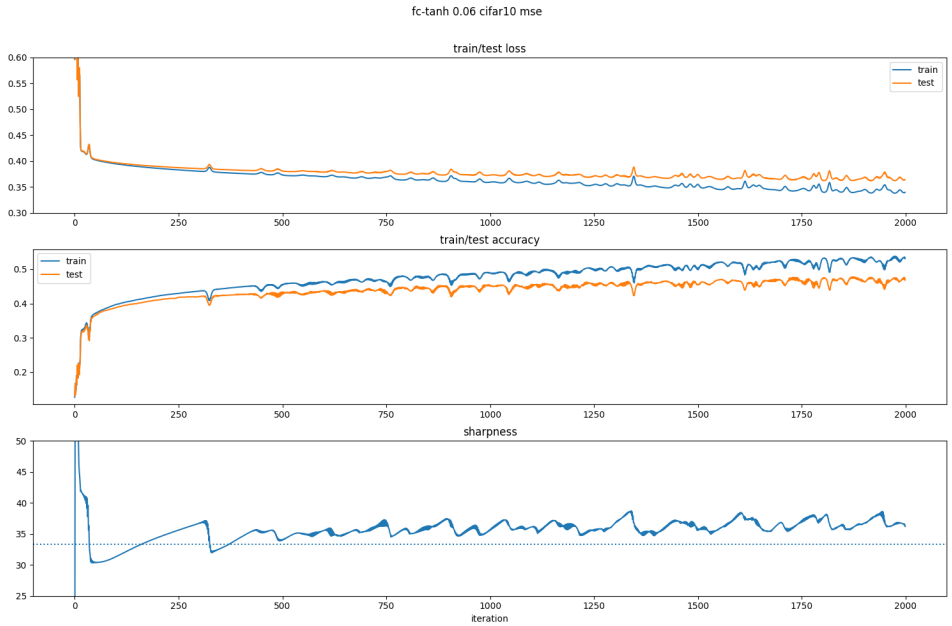

Figure 28: Test and train accuracy and loss for Mean-Squared Error Loss.

L Influencing Factors for *Edge of Stability*: number of parameters

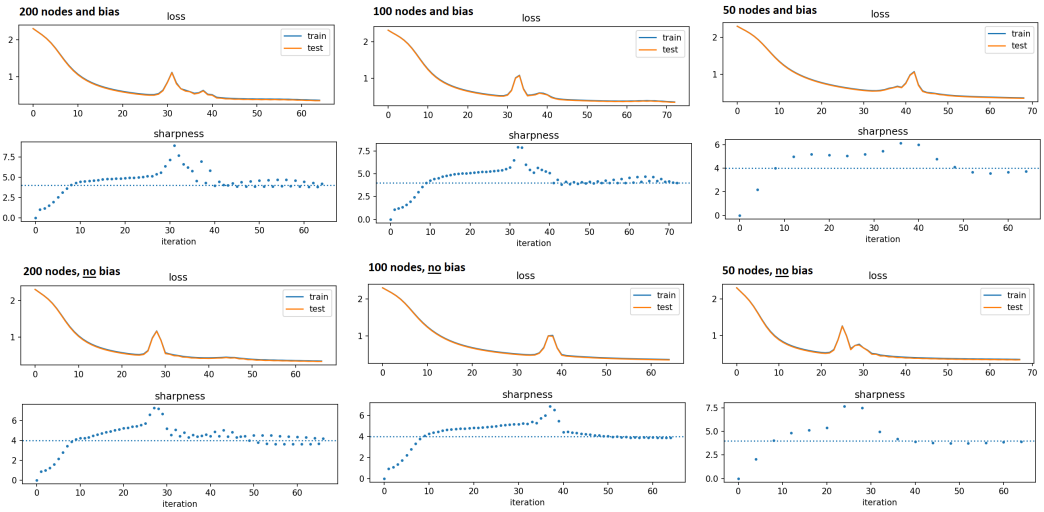

Figure 29: Evolution of loss (top graph) and sharpness (bottom graph) on a neural network with a single hidden layer. The number of nodes in the hidden layer is set to 200 (left), 100 (center) or 50 (right). Presented results considering bias (first two rows) and do not considering it (third and fourth rows). Experiments run using `Config_2` setting presented on Appendix B

M Relation between learning rate, sharpness and generalization

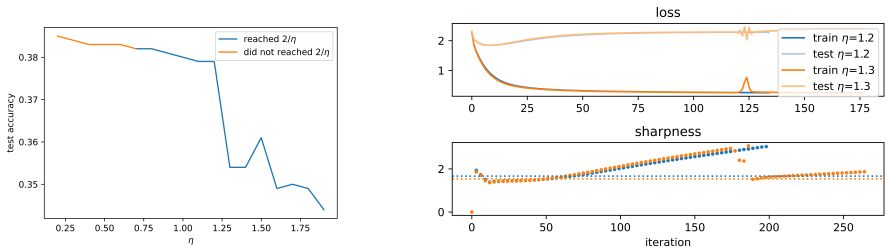

Figure 30: Left: Test accuracy vs learning rate. Right: loss(top) and sharpness(bottom) for $\eta = 1.2$ and $\eta = 1.3$ (bottom right). We observe some relationship between sharpness and the learning rate but further investigation is need. Experiments run using `Config_2` setting presented on Appendix B

In this work we focused on investigating if the generalization is significantly affected immediately when the model enters the edge of stability. If so, the edge of stability threshold could be indicative of overfitting and could be valuable while predicting the performance over unseen samples. As we described, our experiments do not demonstrate the such relationship.

Since the sharpness is related to the adopted learning rate, we discuss some preliminary results regarding the relationship between η and generalization. When evaluating the generalization by the end of training over different learning rates, we have to be cautious in considering different scenarios. With smaller learning rates, the model can be trained without even reaching the threshold and the sharpness can even decrease along the training. When adopting larger learning rates, the progressive sharpness is not observed and the instability occurs with more intensity. The final generalization over different scenarios could not indicate a direct relationship with the sharpness behavior.

Even so, we run experiments considering MNIST and a one hidden layers network. We run the model five times for each learning rate and considered the average of the final testing accuracy. We did observe some possible interesting relationship between the accuracy and the learning rate (see Figure 30). However, for more substantial conclusions a more diverse set of experiments is need. For this reason we did not include such observation on the main report.

On Figure 30, we observe some direct correlation between η and the accuracy on test. Even more, the graph present a big drop in scenarios where there is a decrease of the sharpness in the edge of stability.

N The influence of the learning rate on the Edge of Stability

An increase in learning rate does not appear to substantially alter the shape of the instability in sharpness around the Edge of Stability. In other words, the behavior/amplitude of instability is fairly consistent across different learning rates, whereas the frequency and intensity of instability increases with higher learning rates.

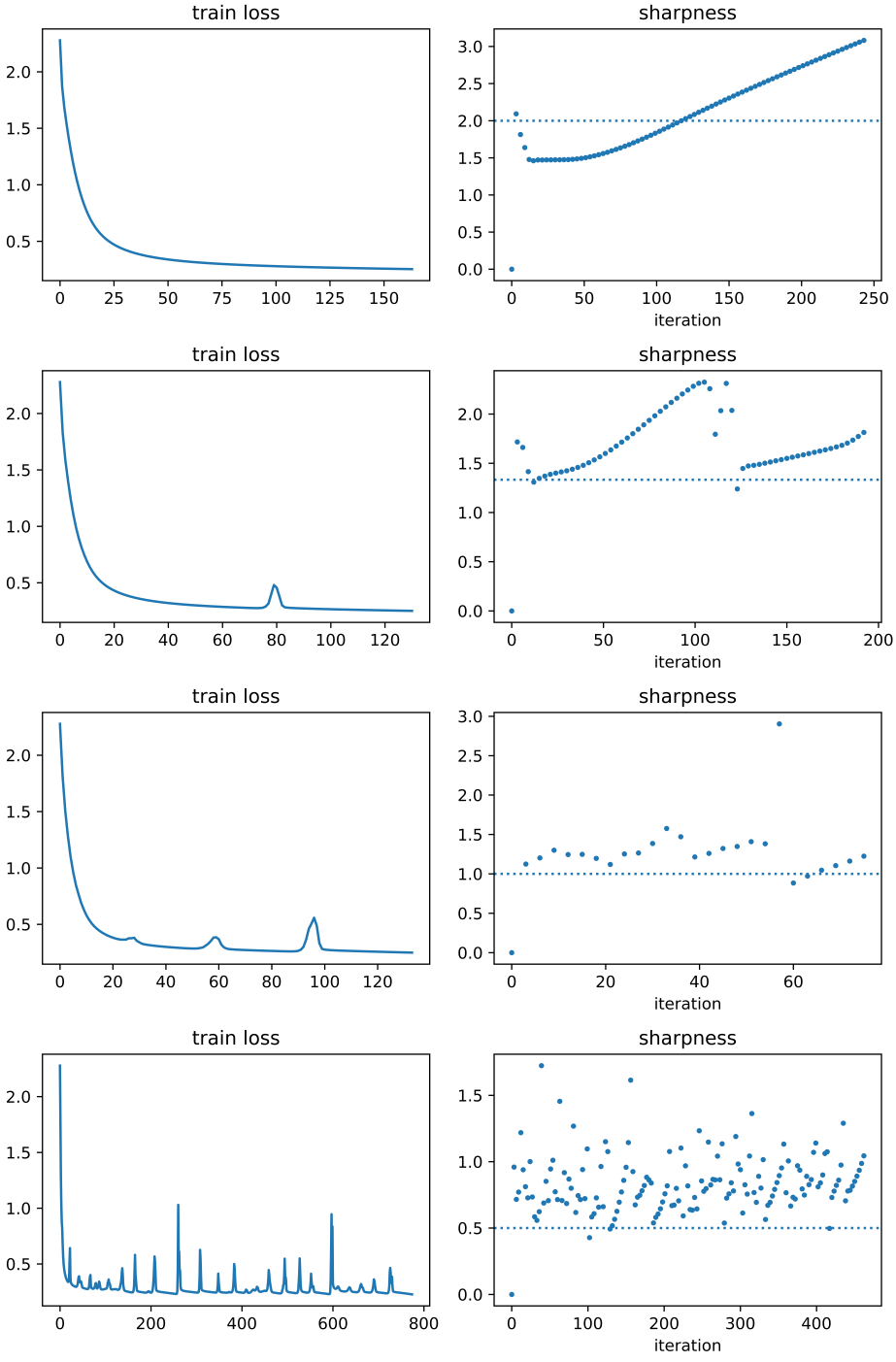

Figure 31: From top to bottom row: $\eta = 0.5$, $\eta = 1$, $\eta = 2$ and $\eta = 4$

O Author Contributions

This section describes the contribution of each author on this work.

Sai Raj Kishore Perla: Background and reasoning behind Progressive Sharpening.

Zhuo Ning: Related work, relationship between sharpness and dataset complexity.

Raphaella Diniz: Relationship between sharpness and model attributes, generalization capacity and instability over each weight.

Ryan Wu: Relationship between sharpness and loss functions, conclusion.

All authors contributed on the discussions, validation of results and in-person presentation.