SIMULATING THE IMPLICIT EFFECT OF LEARNING RATES IN GRADIENT DESCENT

Adrian Goldwaser & Bruno Mlodozeniec & Hong Ge
Department of Engineering
University of Cambridge
Cambridge, UK
{ag2198, bkm28, hg344}@cam.ac.uk

ABSTRACT

Training neural networks can be very sensitive to hyperparameter choice, in particular the learning rate. This choice can often have large effects on final generalisation, not just on training speed and convergence. These two impacts of the learning rate choice are entangled resulting in difficulty finding the true cause of phenomena in deep learning. Building on previous theoretical work, we empirically show how to disentangle these by simulating one learning rate with a smaller one such that the performance remains constant. We show where this method works and what can cause it to break down. We apply this method to the problem of learning rate decay to better understand its effect.

1 INTRODUCTION

Selecting an optimiser and its associated hyperparameters in deep learning is a complex challenge due to their intertwined effects. In traditional optimisation, the choice of an optimiser is usually driven by the convergence speed - the best optimiser is the one that reaches a minimum of the function the fastest. However, in deep learning, this isn't the case. Even when different optimisers achieve the same value for the loss function, they can still lead to models with significantly varying generalisation properties. For example, larger learning rates or smaller batches are often selected not for their speed of convergence but for their perceived ability to produce models that generalise more effectively (Keskar et al., 2017).

The ability to disentangle these effects would be beneficial for design and analysis of deep learning methods. If the generalisation effects of the optimiser hyperparameter choices were made explicit, we could focus solely on optimisation speed when designing new optimisers. We would also be able to examine the generalisation properties of various "implicit" optimiser effects, and the findings would hold independently of the actual optimiser one chooses in practice. Finally, this would allow us to examine the generalisation impact of changes, independently of the impact it might have on viable learning rates from a convergence perspective. In other words, ideally we would be able to study and design for these two criteria independently.

To address this issue, we propose a method to control for the effects of learning rate in stochastic gradient descent. Drawing on the works of Barrett & Dherin (2021) and Smith et al. (2021) — who demonstrated that implicit effects of learning rate and batch size could be represented with an explicit loss term under certain conditions — we show how to empirically account for differing learning rates in real-world settings. We also explore where these ideas break down in practise.

The contributions we make in this paper are as follows:

- We propose a modified loss to simulate one learning rate with another and show its effectiveness in real-world neural networks, finding that the first order approximation always appears sufficient. We show that we can observe this effect in single runs rather than just in expectation as proven in Barrett & Dherin (2021); Smith et al. (2021)
- We show that for deeper networks with residual connections, approximating this using SGD no longer works. We also provide empirical evidence that the entire theory doesn't work well for BatchNorm despite it being used there in the literature.

2 PROBLEM FORMULATION

Barrett & Dherin (2021) show a fascinating result about gradient descent. Consider a loss function $\mathcal{L}(\theta)$ parameterised by $\theta \in \mathbb{R}^p$ trained under gradient descent with learning rate h. That is, $\theta_{k+1} = \theta_k - h\nabla_\theta \mathcal{L}(\theta_k)$. The trajectory follows the same trajectory of gradient flow under a modified loss (up to terms of order $\mathcal{O}(h^3)$). That is, $\dot{\theta}_t = \nabla \mathcal{L}_{\text{GD}}(\theta_t)$ with $\mathcal{L}_{\text{GD}}(\cdot)$ as given below:

$$\mathcal{L}_{\text{GD}}(\theta) = \mathcal{L}(\theta) + \frac{h}{4} \|\nabla_\theta \mathcal{L}(\theta)\|_2^2 + \mathcal{O}(h^3) \tag{1}$$

Smith et al. (2021) extend this result to stochastic gradient descent. Suppose there are $m = \frac{N}{B}$ batches of size B for a dataset of size N. Let $\hat{\mathcal{L}}_k(\cdot)$ be the original loss function restricted to the kth minibatch such that $\hat{\mathcal{L}}_k(\theta) = \frac{1}{B}\sum_{i \in \mathcal{B}_k} \mathcal{L}_i(\theta)$, where $\mathcal{L}_i(\theta)$ is the loss on the ith datapoint and $\mathcal{B}_k$ is the set of indices in the kth minibatch. SGD is equivalent to gradient flow under a new modified loss $\mathcal{L}_{\text{SGD}}(\cdot)$. Note that the $\mathcal{O}(m^3h^3)$ makes this approximation potentially break down once the batch size is too small. This is also only shown *in expectation* over all non-overlapping minibatches $\{\mathcal{B}_1, \mathcal{B}_2, \dots \mathcal{B}_m\}$. There are no results we know of which show how consistent this is in practise. As such, no claims can be made about individual runs. The modified loss is as follows:

$$\mathcal{L}_{\text{SGD}}(\theta) = \mathcal{L}(\theta) + \frac{h}{4m} \sum_{k=1}^{m} \|\nabla_\theta \mathcal{L}_k(\theta)\|_2^2 + \mathcal{O}(m^3h^3) \tag{2}$$

Or alternatively:

$$\mathcal{L}_{\text{SGD}}(\theta) = \mathcal{L}_{\text{GD}}(\theta) + \frac{h}{4m} \sum_{k=0}^{m-1} \|\nabla_\theta \mathcal{L}_k(\theta) - \nabla_\theta \mathcal{L}(\theta)\|_2^2 + \mathcal{O}(m^3h^3) \tag{3}$$

Building on Equation (1) we propose a loss function, $\widetilde{\mathcal{L}}(\theta) = \mathcal{L}(\theta) + R(\theta)$, and show how it cancels out the extra terms in $\mathcal{L}_{\text{GD}}(\cdot)$ in order to be equivalent to running gradient descent with a new learning rate η. Throughout this paper, h represents the true learning rate (following previous work) and η represents the learning rate we wish to simulate. We use $\lambda = \eta - h$ to make the derivations below cleaner.

First note that for any $R(\theta)$, we have that, $\widetilde{\mathcal{L}_{\text{GD}}}(\theta) \approx \mathcal{L}(\theta) + R(\theta) + \frac{h}{4}\|\nabla_\theta \mathcal{L}(\theta)\|_2^2 + \frac{h}{4}\|\nabla_\theta R(\theta)\|_2^2 + \frac{h}{2}\nabla_\theta \mathcal{L}(\theta)^\top \nabla_\theta R(\theta)$. Our proposed addition to the loss function is:

$$R(\theta; h, \eta) = \frac{\lambda}{4}\|\nabla_\theta \mathcal{L}(\theta)\|_2^2 - \frac{h\lambda}{4} Z(\theta) \tag{4}$$

$$R(\theta; h, \eta) = \frac{\eta - h}{4}\|\nabla_\theta \mathcal{L}(\theta)\|_2^2 - \frac{h(\eta - h)}{4} Z(\theta) \tag{5}$$

$$Z(\theta) = \nabla_\theta \mathcal{L}(\theta)^\top \nabla_\theta^2 \mathcal{L}(\theta) \nabla_\theta \mathcal{L}(\theta) \tag{6}$$

Where $Z(\theta) = \nabla_\theta \mathcal{L}(\theta)^\top \nabla_\theta^2 \mathcal{L}(\theta) \nabla_\theta \mathcal{L}(\theta)$. Note that $R(\theta)$, $Z(\theta)$ and their derivatives can be calculated efficiently using higher order autodiff and Jacobian-vector products. For example $Z(\theta)$ is the inner product of $\nabla_\theta \mathcal{L}(\theta)$ and $\nabla_\theta \|\nabla_\theta \mathcal{L}(\theta)\|_2^2$ — both of which can be calculated using higher order autodiff. If we consider the modified loss function we would need to follow under gradient flow to achieve the same trajectory using Equation (1), we get the following:

$$\widetilde{\mathcal{L}_{\text{GD}}}(\theta) \approx \mathcal{L}(\theta) + \frac{h + (\eta - h)}{4}\|\nabla_\theta \mathcal{L}(\theta)\|_2^2 - \frac{h\lambda}{4} Z(\theta) + \frac{h}{4}\|\nabla_\theta R(\theta)\|_2^2 + \frac{h}{2}\nabla_\theta \mathcal{L}(\theta)^\top \nabla_\theta R(\theta) \tag{7}$$

$$= \mathcal{L}(\theta) + \frac{h + \lambda}{4}\|\nabla_\theta \mathcal{L}(\theta)\|_2^2 + \mathcal{O}(\lambda^2 h + h^2 \lambda) \tag{8}$$

$$\approx \mathcal{L}(\theta) + \frac{h + \lambda}{4}\|\nabla_\theta \mathcal{L}(\theta)\|_2^2 \tag{9}$$

Now to simulate a learning rate of η we can set $\lambda = \eta - h$. We can also set $\lambda = -h$ to simulate gradient flow. This gives us the regulariser above, now in terms of η. $R(\theta; h, \eta) = \frac{\eta-h}{4} \|\nabla_\theta \mathcal{L}(\theta)\|_2^2 - \frac{h(\eta-h)}{4} Z(\theta)$. There is a first order approximation (in h) of this regulariser as simply, $R(\theta) = \frac{\eta-h}{4} \|\nabla_\theta \mathcal{L}(\theta)\|_2^2$. This ignores terms of order $\mathcal{O}(h^2)$ and beyond. We will investigate both of these in the experiments.

2.1 An aside on BatchNorm

Many neural networks in computer vision use BatchNorm (Ioffe & Szegedy, 2015) while most uses in transformers instead rely on LayerNorm (Ba et al., 2016). Close to state-of-the-art results in computer vision tasks can often be achieved with LayerNorm however (Liu et al., 2022). Similarly there has been work on modifying BatchNorm for use in transformers (Shen et al., 2020). As such it seems likely that the difference between them in terms of performance is smaller than might be expected and likely could go either way depending on the effort put into each.

There is one very important distinction however. When using BatchNorm, the batch loss is no longer the average of datapoint losses. This fact makes a large amount of theoretical analyses go out the window, including the SGD results of Smith et al. (2021). Another example of this weirdness of BatchNorm is that there are initialisation settings that stay on the "edge of chaos" in very deep networks for most architectures studied (Schoenholz et al., 2016; Yang & Schoenholz, 2017) except for BatchNorm (Yang et al., 2019). In our case we show empirically in Section 3.2 that the theory from Barrett & Dherin (2021) does not appear to hold for BatchNorm either in practical settings, though the full batch theory would still appear to hold.

3 Experiments

We investigate the loss function addition term, $R(\theta) = \frac{\eta-h}{4} \|\nabla_\theta \mathcal{L}(\theta)\|_2^2$, in order to simulate one learning rate (η, the *simulated LR*) with another (h, the *true LR*). We compare maximum test accuracy across the run (i.e. optimal early stopping) as the scaling of number of steps to train may not be exactly the ratio of learning rates. Section 3.2 shows that this scaling is in fact not accurate, requiring more careful scaling of epochs.

3.1 Basic CNN

We begin by looking at the accuracy of training basic CNN on CIFAR10 *under SGD* with the full $R(\cdot)$ from Equation (4). This includes the second order terms. Results are shown in Figure 1. The left graph shows all runs and the right graph is filtered only to those where $\eta \geq h$ — i.e. where the simulated learning rate is at least the true learning rate. It is clear that simulating of smaller learning rates does not work, most of these runs diverged. This is because there is a negative regularisation term ($h - \eta < 0$) such that there is a pressure to make the gradient as large as possible. The exception to this is that with a very small learning rate, gradient flow can be simulated. This can already give a good idea of when a learning rate is small enough to approximate gradient flow, which is useful in its own right.

For the rest of the experiments, we consider only the cases where $\eta \geq h$ as shown in Figure 1 (right). Here we see the expected results with each series representing the same simulated learning rate and resulting in straight lines.

We next consider ignoring the second order term and examine the effects. We see in Figure 6[1] that this is almost exactly the same as the results with the second order term included. This is not surprising as the second order terms get very small and in practise are unlikely to modify the trajectory in any meaningful way. As such we use the first order approximation where possible.

In order to better understand that this is following a similar trajectory, we adopt a very simple scaling strategy and plot training curves of runs with the same simulated learning rate. We scale the step by the true learning rate (which we call *time*). We plot these in Figure 2. The performance curves overlap extremely well. Larger trrue learning rates result in more variability and instability during

[1] We move this figure to Appendix A for brevity as it is visually identical to Figure 1

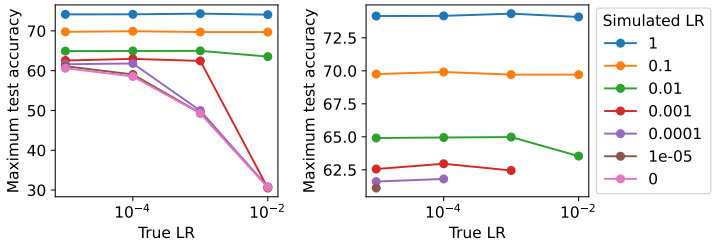

Figure 1: This figure shows the test accuracy under optimal early stopping for a variety of true and simulated learning rates. The right graph is identical except with all runs where the simulated learning rate is less than the true learning rate removed. We do this to clean up the graph as those runs tend to diverge. True learning rates of 0.1 and 1 diverge and are not included.

training but the overall structure is similar. For a simulated LR of 0.01 (top right), we note a slight difference with a true LR of 0.01 not reaching quite as high as when simulating with a smaller LR. This was also visible, but less noticeable, in Figures 1 and 6. We believe that this is due to 0.01 being on the border of how large the learning rate can be to converge on this model. As such we get even more instability resulting in worse performance. We also include a control of the same graph but without using Equation (4) as an addition to the loss. We can see that the trajectories are significantly more different here.

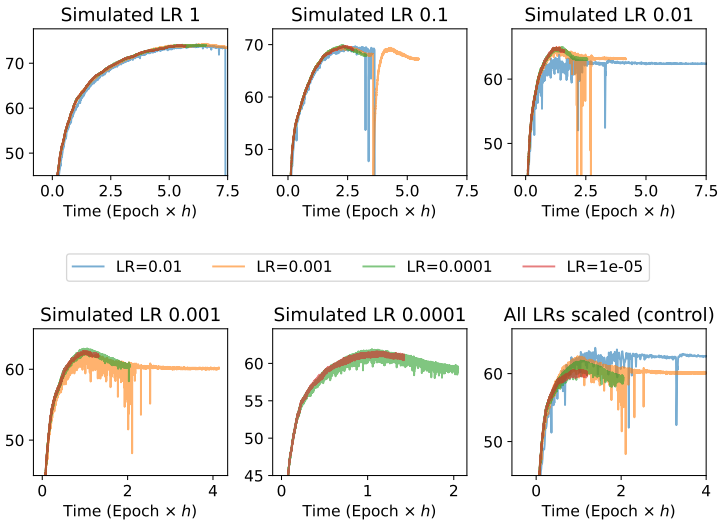

Figure 2: The first five graphs show the similarities in the test performance during training for runs with the same simulated learning rate. The x axis is scaled by the true learning rate of the run to make them line up. The final (bottom right) graph shows the same procedure without any learning rate simulation as a control.

3.2 ResNet

In order to extend the empirical evidence, we also run this with a larger model. We chose ResNet18 as it adds two more common architectural elements; residual connections and normalisation. It is also a much deeper model than the simple CNN we used in the previous section. We consider a ResNet18 with the BatchNorm replaced by LayerNorm. We do this because BatchNorm can have a number of issues with theoretical results as explained in Section 2.1. We also show BatchNorm in Figure 3 to show the issues associated. The absolute accuracies are low here due to a combination

of lack of data augmentation, weight decay and low learning rate. However, we mostly care about relative accuracies. We also ran the full batch BatchNorm results from Figure 3 with simulating a learning rate of 1 instead. This gave test accuracies of 65.0%, 65.9%, 63.3% and 59.1% with true learning rates of 1, 0.1, 0.01, 0.001 respectively. While these are better accuracies, the learning rate simulation still does not hold.

We do not see this as a large issue for the applicability of our technique. Most modern architectures use LayerNorm instead of BatchNorm, including almost all transformers and state-of-the-art CNNs (Liu et al., 2022).

We do have to revert to full batch gradient descent however as here the stochasticity term from Equation (2) begins to effect results. The first-order approximation does not affect the applicability of SGD here. We show the difference between full batch GD and minibatch SGD in Figure 3. There are three important insights from this graph.

1. The minibatch SGD approximation breaks down here due to some combination of residual layers and depth
2. This happens when the SGD performance is different to the fullbatch performance (unlike in Figure 6)
3. Performance *improves* as the true learning rate goes down if using SGD

Similarly, the scaling method used for a basic CNN no longer works; instead, we scale by lining up the point of maximum test accuracy. This location has already been looked at in Barrett & Dherin (2021) as an important learning rate-independent location. We show the difference between these in Figure 4

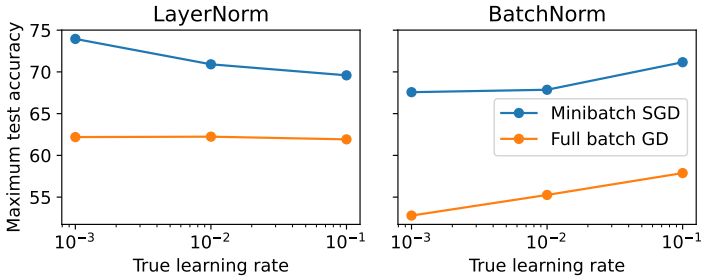

Figure 3: Here we show the comparison between full batch and mini-batch simulation for ResNet18 with both LayerNorm and BatchNorm. All runs are simulating a learning rate of 0.1. Of the four, only full batch simulation with LayerNorm remains constant. The theorems of Barrett & Dherin (2021) only hold in the full batch case.

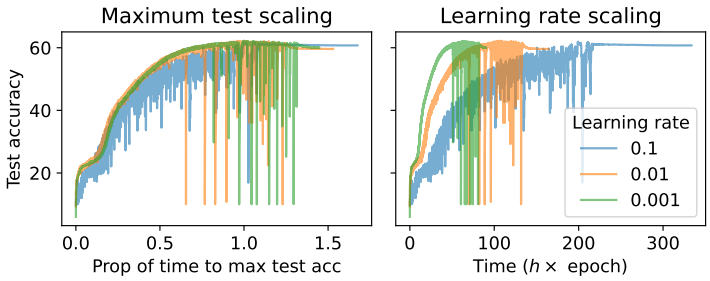

Figure 4: This compares two scaling methods to make the training curves line up. On the left the point with maximum test accuracy is found and the proportion of steps towards that point is used. The right graph simply scales with the true learning rate as in Figure 2 which no longer works.

Table 1: Learning rate decay comparison

Initial LR	LR decay type	Test accuracy (± std error)
0.05	Decay both	65.57% (± 0.14)
	Decay LR	65.84% (± 0.14)
	No decay	65.73% (± 0.13)
0.01	Decay both	63.84% (± 0.13)
	Decay LR	64.72% (± 0.13)
	No decay	64.07% (± 0.13)

We include a version of Figures 3 and 4 with second order terms in Appendix A for reference, but the story is similar as the issues are caused by SGD or by BatchNorm.

3.3 Understanding learning rate decay

A common practise in deep learning optimisation is to decay the learning rate. However, the learning rate has two impacts on optimisation. The first is with regards to optimisation speed versus precision, and the second relates to the modified loss function being optimised. This raises the interesting question of which of these is important in learning rate decay. With the new tool that we have developed, we can now approach this problem. We do this by comparing three learning rate decay schemes. We summarise these schedules below and in Figure 5.

1. No decay: neither true nor simulated LR decays
2. Decay LR: decay the true LR while keeping the simulated LR constant
3. Decay both: decay both the true and simulated LR

We run these for a two initial learning rates with the setting from Section 3.1 and show the results of this in Table 1. Running across 50 random seeds, we can see a slight improvement for only decaying the true LR, keeping the simulated LR constant. For the larger initial LR, these are within error margins but for the smaller one they are significant. This provides some evidence that we ideally want to use a smaller LR to help converge but not remove the useful implicit regularisation of a larger learning rate. Although this is a limited setting with a small effect size, it is a clear example of how this technique can be applied to give novel insights.

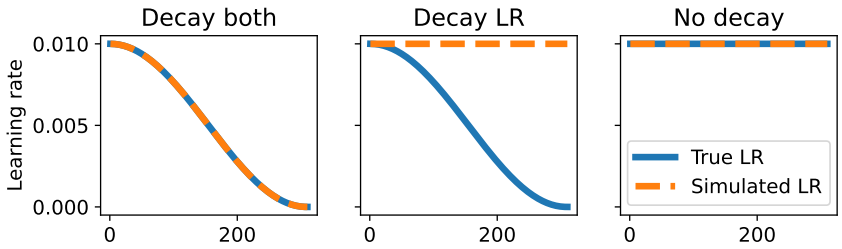

Figure 5: Visual depiction of the three learning rate schedules. Left: both true LR and simulated LR decay together. Centre: true LR decays but simulated LR remains constant. Right: Neither true nor simulated LR decay.

4 Related work

This work builds directly off of Barrett & Dherin (2021) which derived and proved Equation (1) to help explain the impact of the learning rate in gradient descent on generalisation. Smith et al. (2021) did the same for SGD, proving Equations (2) and (3). Both works also include initial empirical experiments using the derived loss as an explicit regulariser and find it helpful.

Novack et al. (2023) empirically examine the impact of a number of theories about the implicit regularisation of SGD, including Smith et al. (2021). They do this through *micro-batches* to avoid the intrinsic reliance on full-batch evaluation required. While they find that this can work, the substitution of micro-batches causes the results to not be equivalent rendering it unhelpful for our goal of fully simulating one hyperparameter setting with another.

All of these works use ResNets with BatchNorm as part of their experimentation, where much of the theory has not been proven to hold. We do not know if this had as large an impact on their results as it did on ours, however we conjecture that their results would show stronger alignment with theory if this was replaced by LayerNorm.

The impact of learning rate on convergence guarantees and rates has been studied extensively in the field of optimisation, some work has also looked at its impact on generalisation. Cohen et al. (2021) look at the how the sharpness of the loss changes throughout training based on the learning rate, applying the common hypothesis that flatter minima tend to generalise better. Many studies look at trying to classify learning rates into different regimes where they behave quantitatively differently (Kong & Tao, 2020; Lewkowycz et al., 2020). Wang et al. (2022) classify the exact implicit bias of large LR gradient descent in the limited setting of matrix factorisation. We believe that if we can show the claims of this paper then papers with insights such as those mentioned can be run with more controlled experiments to gain confidence on both theories. Additionally, it lets us separate out the optimisation concerns (convergence) from the generalisation ones (implicit bias).

5 Conclusion

We have shown a way to simulate larger learning rates with smaller ones in order to be able to run more controlled experiments. This method works even when approximated with just the first order terms. We show that this works well to simulate larger learning rates on a single run, rather than just in expectation as shown theoretically in Barrett & Dherin (2021). We show that either full batch or an estimator is required for larger models with residual connections.

We also raise a note that much of the underlying theory does not work on architectures with BatchNorm, despite the regulariser being used on such networks with positive results in the literature. We show empirically that even in the full batch case, BatchNorm does not appear to work for learning rate simulation.

We apply this method to understanding learning rate decay, finding that while decreasing the LR can improve convergence, there is a trade-off due to it also removing useful implicit regularisation. We show that this can be mitigated through the addition of the regularisation term.

5.1 Limitations and future work

The most important hole to fill is to run these experiments on larger datasets and models to confirm that this trend we have shown holds more broadly.

The second limitation is the requirement for full batch to ensure correctness. We require full batch runs to correctly, but using the results from Smith et al. (2021), we believe this can be addressed using a Monte Carlo estimator.

Finally, all of this work can easily be generalised to batch size once the above two points are addressed. This would allow simulation of arbitrary batch sizes and learning rates to allow more complex controlled experiments.

Acknowledgments

Removed for double-blind review

References

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016. URL http://arxiv.org/abs/1607.06450.

David G. T. Barrett and Benoit Dherin. Implicit gradient regularization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=3q5IqUrkcF.

Jeremy Cohen, Simran Kaur, Yuanzhi Li, J. Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=jh-rTtvkGeM.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 448–456. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/ioffe15.html.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=H1oyRlYgg.

Lingkai Kong and Molei Tao. Stochasticity of deterministic gradient descent: Large learning rate for multiscale objective function. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1b9a80606d74d3da6db2f1274557e644-Abstract.html.

Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism. *CoRR*, abs/2003.02218, 2020. URL https://arxiv.org/abs/2003.02218.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 11966–11976. IEEE, 2022. doi: 10.1109/CVPR52688.2022.01167. URL https://doi.org/10.1109/CVPR52688.2022.01167.

Zachary Novack, Simran Kaur, Tanya Marwah, Saurabh Garg, and Zachary Chase Lipton. Disentangling the mechanisms behind implicit regularization in SGD. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=LE5LxBgjB4V.

Samuel S. Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. *CoRR*, abs/1611.01232, 2016. URL http://arxiv.org/abs/1611.01232.

Sheng Shen, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. Powernorm: Rethinking batch normalization in transformers. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8741–8751. PMLR, 2020. URL http://proceedings.mlr.press/v119/shen20e.html.

Samuel L. Smith, Benoit Dherin, David G. T. Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=rq_Qr0c1Hyo.

Yuqing Wang, Minshuo Chen, Tuo Zhao, and Molei Tao. Large learning rate tames homogeneity: Convergence and balancing effect. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=3tbDrs77LJ5.

Greg Yang and Samuel S. Schoenholz. Mean field residual networks: On the edge of chaos. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 7103–7114, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/81c650caac28cdefce4de5ddc18befa0-Abstract.html.

Greg Yang, Jeffrey Pennington, Vinay Rao, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. A mean field theory of batch normalization. *CoRR*, abs/1902.08129, 2019. URL http://arxiv.org/abs/1902.08129.

A Experimental details and further experiments

We include here the details of the experiments. We also include three additional plots which swap whether the second order term in Equation (4) are included. These graphs are almost identical to their counterparts so we keep them in this appendix for clarity. Figure 6 is a version of Figure 1 without the second order terms. Figures 7 and 8 are the equivalents of Figures 3 and 4, respectively, with the second order terms added.

Experiments are run on cifar10 and use either a basic CNN (two 5x5 convolutional layers with 50 channels with average pooling followed by a dense layers of sizes 256, all with ReLU activations) or a ResNet18 (sometimes replacing BatchNorm with LayerNorm). Full batch experiments limit cifar10 to 30k train datapoints to allow the entire dataset to fit in GPU memory.

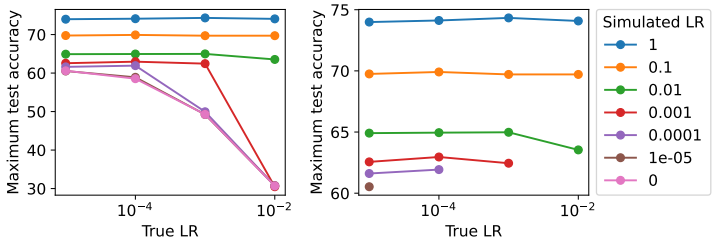

Figure 6: This is a copy of Figure 1 *without* the second order terms.

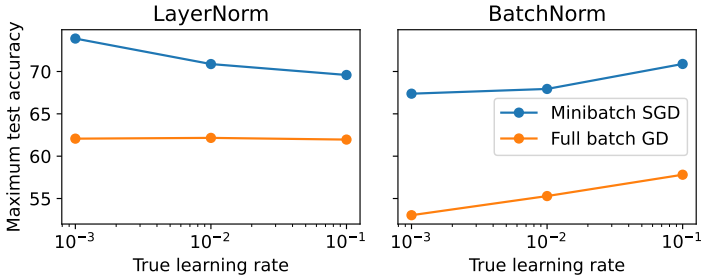

Figure 7: This is a copy of Figure 3 with second order terms included.

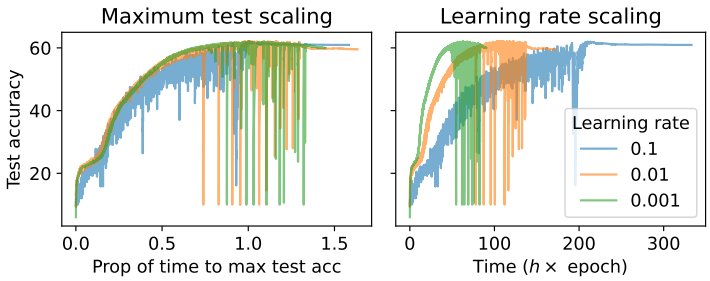

Figure 8: This is a copy of Figure 4 with second order terms included.