On the Maximum Hessian Eigenvalue and Generalization

Anonymous Author(s)
Affiliation
Address
email

Abstract

The mechanisms by which certain training interventions, such as increasing learning rates and applying batch normalization, improve the generalization of deep networks remains a mystery. Prior works have speculated that "flatter" solutions generalize better than "sharper" solutions to unseen data, motivating several metrics for measuring flatness (particularly $\lambda_{\max}$, the largest eigenvalue of the Hessian of the loss); and algorithms, such as *Sharpness-Aware Minimization* (SAM) [1], that *directly* optimize for flatness. Other works question the link between $\lambda_{\max}$ and generalization. In this paper, we present findings that call $\lambda_{\max}$'s influence on generalization further into question. We show that: (1) while larger learning rates reduce $\lambda_{\max}$ *for all batch sizes*, generalization benefits sometimes vanish at larger batch sizes; (2) by scaling batch size and learning rate simultaneously, we can change $\lambda_{\max}$ without affecting generalization; (3) while SAM produces smaller $\lambda_{\max}$ *for all batch sizes*, generalization benefits (also) vanish with larger batch sizes; (4) for dropout, excessively high dropout probabilities can degrade generalization, even as they promote smaller $\lambda_{\max}$; and (5) while batch-normalization does not consistently produce smaller $\lambda_{\max}$, it nevertheless confers generalization benefits. While our experiments affirm the generalization benefits of large learning rates and SAM for minibatch SGD, the GD-SGD discrepancy demonstrates limits to $\lambda_{\max}$'s ability to explain generalization in neural networks.

1 Introduction

Researchers have devised a repertoire of practices that reliably improve the generalization error of neural networks [2, 3, 4, 5, 6, 7, 8, 9, 10]. Techniques linked to improved test accuracy across a wide variety of datasets and architectures include (i) small batch sizes for minibatch SGD; (ii) larger learning rates for minibatch SGD; (iii) batch normalization; and (iv) dropout; among others [5, 7, 8, 9, 10]. Absent of guiding theory to explain *why* these training interventions improve generalization, researchers are left to rely on intuition. One popular claim in the literature is that *flat minima* generalize better than *sharp* (non-flat) *minima* [11, 10]. Thus motivated, Foret et al. [1] proposed *Sharpness-Aware Minimization* (SAM), which directly optimizes for flat solutions.

While there are many mathematical definitions of flatness, a common one is the largest eigenvalue of the Hessian of the loss with respect to the model parameters on the training set, which we denote $\lambda_{\max}$ [9, 12, 13]. Already in the literature, there are indications that $\lambda_{\max}$ may not be causally linked with improved generalization. In particular, Dinh et al. [13] showed that given any set of weights for a ReLU network, there exists a new set of weights that produce the same function, but have arbitrarily high $\lambda_{\max}$. However, a motivated reader might still argue that Dinh et al. [13]'s construc-

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

tion is artificial: perhaps standard optimizers and initializations do not give rise to pathologically sharp solutions. Thus, while Dinh et al. [13] convincingly demonstrates that "high $\lambda_{\max}$" does not causally imply "bad generalization," it leaves open the possibility that $\lambda_{\max}$ might still be a useful predictor of generalization in realistic deep learning scenarios.

In this paper, we conduct an experimental study to investigate the association between $\lambda_{\max}$ and generalization. Our results call the relationship between $\lambda_{\max}$ and generalization further into question. We provide four examples of training interventions that promote smaller values for $\lambda_{\max}$, but either do not change, or even degrade, generalization. First, we show that for a non-batch-normalized VGG network *in the large-batch setting*, increasing the learning rate reliably results in a smaller $\lambda_{\max}$, yet does not improve generalization. Second, we show that for the same architecture in the *small-batch setting*, increasing the learning rate and batch size in tandem while preserving their ratio reliably results in a smaller $\lambda_{\max}$, yet leaves generalization unchanged (due to the linear scaling rule [14]). Third, we show that for all batch sizes, a higher sharpness penalty in the context of SAM produces smaller $\lambda_{\max}$. However, SAM's generalization benefits diminish as batch size increases, vanishing entirely in the large-batch setting. (In line with Foret et al. [1], we observe that in the *small-batch* setting, SAM promotes both flatness and generalization.) Fourth, we find that while dropout promotes smaller $\lambda_{\max}$, excessively high dropout probabilities degrade generalization. In these four instances, promoting smaller $\lambda_{\max}$ does not reliably boost generalization. Our results suggest that $\lambda_{\max}$ may not be reliably correlated with generalization in practice.

We also provide an example of an intervention—batch-normalization—that improves generalization, without changing $\lambda_{\max}$. In particular, we find that while adding batch-normalization does not result in smaller $\lambda_{\max}$ in the large-batch setting, it does confer generalization benefits at large learning rates. This finding further calls into question the correlation of $\lambda_{\max}$ with generalization.

2 Background and Related Work

In deep learning, certain design choices, such as batch size and learning rate, are known to have an implicit effect on generalization [9, 5, 10]. These choices are believed to implicitly guide the training process towards solutions with favorable geometric properties [9, 15, 12]. These favorable geometric properties are broadly referred to as "flatness" (versus "sharpness").

Metrics for flatness Some metrics, such as the maximum Hessian eigenvalue, measure the *worst-case* loss increase under an *adversarial* perturbation to the weights [10, 16], while other proposed metrics, such as the Hessian trace, measure the *expected* loss increase under *random* perturbations to the weights. Finally, there are volume-based metrics, which can be defined as a large connected region in the weight space where the error remains approximately constant [11]. In this paper, we focus exclusively on the leading Hessian eigenvalue $\lambda_{\max}$ and its relationship to generalization.

$\lambda_{\max}$ as a metric for flatness The maximum Hessian eigenvalue $\lambda_{\max}$ is commonly viewed as a potential sharpness/flatness metric [17, 10, 18, 9]. For example, when evaluated at a global or local minimum, the sharpness metric from Keskar et al. [10] is equivalent to a scaled version of $\lambda_{\max}$, provided that the training objective is well-approximated by its second-order Taylor approximation [10]. Lewkowycz et al. [12] argued that SGD with large learning rates generalizes well because dynamical instability at initialization causes the optimizer to "catapult" to regions where the maximum eigenvalue of the neural tangent kernel and of the Hessian are smaller. Foret et al. [1] proposed Sharpness-Aware Minimization (SAM), aiming to improve generalization by *explicitly* penalizing a sharpness metric, and has grown in popularity [19, 20]. While the final SAM procedure (their Algorithm 1) does not directly penalize $\lambda_{\max}$, the idealized regularizer in their Equation 1 is, at a local or global minimum, equivalent to the maximum Hessian eigenvalue, if one assumes that the training objective is well-approximated by its second-order Taylor approximation. When *not* at a local or global minimum, the maximum Hessian eigenvalue can be used to drive an upper bound on the idealized SAM regularizer (see Appendix A).

Known issues with generalization metrics based on flatness Dinh et al. [13] identified the following issues with sharpness-based metrics. Given a ReLU network, one can rescale the weights

without affecting generalization (or even the function) [13]; this rescaling takes advantage of the fact that for e.g. a one hidden layer ReLU network with parameters (θ_1, θ_2), $\{(\alpha\theta_1, \alpha^{-1}\theta_2), \alpha > 0\}$ is an infinite set of observationally equivalent parameters with arbitrarily high sharpness as $\alpha \to \infty$. Thus, Dinh et al. [13] theoretically showed that large $\lambda_{\max}$ (i.e., sharp minima) are not necessarily bad for generalization. By the same token, Granziol [21] empirically demonstrated that L2 regularization can lead to minima with larger $\lambda_{\max}$, which generalize well. Zhang et al. [22] showed that the correlation between the Hessian spectral norm (which is closely related to $\lambda_{\max}$) and generalization breaks down when switching the optimizer from SGD to Adam. We now discuss how our findings differ from the aforementioned works. First, the counterexample in [13] is artificial: these reparameterized models are unnatural and would likely not be reached by SGD. Given the empirical nature of our experiments, our flat minima are reachable by SGD. Moreover, we do not manipulate the model parameters and still show limitations to the connection between $\lambda_{\max}$ and generalization. Next, unlike the negative examples in [21, 22], our settings are simple (i.e., no learning rate schedules) and hold when SGD is the optimizer. And, we provide *multiple* ways to control $\lambda_{\max}$, beyond L2 regularization [21]. Given the simplicity of our experiments and the number of settings we provide, our work demonstrates that limitations to the connection between $\lambda_{\max}$ and generalization are widespread, rather than subject to few caveats [13, 21, 22]. As such, our work highlights the importance of understanding the relationship between $\lambda_{\max}$ and generalization, and questions whether $\lambda_{\max}$ should be treated as a generalization metric at all.

Large learning rates lead to smaller $\lambda_{\max}$ Several prior works have observed that training with larger learning rates causes $\lambda_{\max}$ to be smaller along the optimization trajectory [9, 23, 24]. In at least the special case of full-batch gradient descent, this relationship is due to dynamical instability [12, 24]: gradient descent with learning rate η will rapidly escape from any region of the loss landscape where $\lambda_{\max}$ exceeds the value $2/\eta$, and so in practice, gradient descent with learning rate η spends most of training in regions of the loss landscape where $\lambda_{\max} \approx 2/\eta$ [24]. In the more general case of stochastic gradient descent, the association between large learning rates and small $\lambda_{\max}$ has also been hypothesized to be related to dynamical instability [25, 23].

Linear scaling rule The linear scaling rule states that in the small-batch regime, one can maintain generalization performance when increasing batch size in minibatch SGD by holding fixed the ratio of batch size to learning rate [14, 26, 27, 9]. Jastrzebski et al. [9] finds that smaller values of the batch size to learning rate ratio correlate with improved generalization and small $\lambda_{\max}$, which is consistent with the popular intuition that small batch sizes (and large learning rates) generalize better by guiding SGD towards "flatter" solutions [9, 15, 12].

3 Experiments

In this section, we provide four examples[1] of training interventions that promote smaller values for $\lambda_{\max}$, but either do not change, or degrade, generalization. We then provide one example of an intervention that improves generalization, without changing $\lambda_{\max}$[2].

3.1 Scaling learning rates in the small- and large-batch regimes

We demonstrate that when a VGG-11 without batch normalization is trained on CIFAR-10 in the large-batch setting, larger learning rates cause $\lambda_{\max}$ to be smaller, yet do not promote better generalization. (By contrast, in the *small-batch* setting, larger learning rates both reduce $\lambda_{\max}$ and improve generalization.) For the plots in Figure 1, we train with minibatch SGD until reaching 99% train accuracy. We consider various learning rates and both the small-batch (red) and large-batch (blue) regime. In our experiments, we use small batch size 100 and large batch size 5000. For *small-batch SGD*, we make the following observations.

(i) For small-batch SGD, large learning rates generalize better We plot the test accuracy attained by small-batch SGD at varying learning rates (Figure 1). Observe that small-batch SGD generalizes better when the learning rate is larger, as observed previously [9, 5].

[1] We use the exact same weight initializations (i.e. the same seeds) in each experimental setting.

[2] Due to compute constraints, we calculate $\lambda_{\max}$ over a subset of the training dataset.

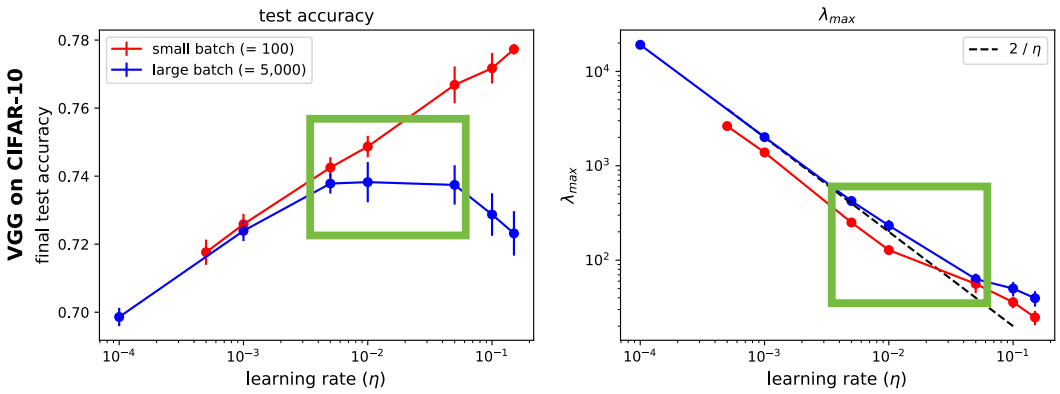

Figure 1: **Small $\lambda_{\max}$ do not always generalize better.** We train a VGG11 (no BN) using SGD at a range of learning rates. We plot the final test accuracy and $\lambda_{\max}$ (averaged over 5 runs with different random seeds) as a function of the learning rate. Observe that as the learning rate increases, the final test accuracy increases for small-batch and plateaus for large-batch SGD. In both cases, as the learning rate increases, $\lambda_{\max}$ decreases.

(ii) For small-batch SGD, large learning rates find small $\lambda_{\max}$ We plot $\lambda_{\max}$ attained by small-batch SGD at varying learning rates and observe that $\lambda_{\max}$ is smaller when the learning rate is large (Figure 1). These findings also have precedents in the literature [9, 5].

Moreover, these two findings are consistent with the popular hypothesis that a small $\lambda_{\max}$ is the *causal* mechanism by which large learning rates promote generalization. However, while this hypothesis is consistent with experimental data for *small-batch* SGD, we will see that it is *not* consistent with experimental data for large-batch or full-batch SGD.

(iii) For large-batch SGD, large learning rates still find small $\lambda_{\max}$ As with *small-batch* SGD, large learning rates also find solutions with small $\lambda_{\max}$ for *large-batch* SGD (Figure 1). The mechanism behind this phenomenon is well-understood: as [24] showed, full-batch gradient descent typically operates in a regime (the Edge of Stability), in which $\lambda_{\max}$ constantly hovers just above the numerical value $\frac{2}{\eta}$, where η is the learning rate. Thus, for large-batch SGD, $\lambda_{\max}$ at the gradient descent solution is approximately $\frac{2}{\eta}$. To confirm this, we draw a dotted black line to mark the line $\frac{2}{\eta}$; we can see that $\lambda_{\max}$ of the large-batch SGD solution lies approximately on the line.

(iv) But, for large-batch SGD, large learning rates don't generalize better Beyond some point, large learning rates **do not** generalize better than small learning rates for large-batch SGD (Figure 1). Thus, in this experiment, smaller $\lambda_{\max}$ is not always correlated with better generalization. For further discussion, please see Appendix B.1.

3.2 Sharpness-Aware Minimization can degrade generalization for large batch sizes

Motivated by the intuition that flatness causally promotes generalization, Foret et al. [1] proposed an algorithm, Sharpness-Aware Minimization (SAM), which aims to improve generalization by directly penalizing a sharpness metric. Their idealized sharpness metric, given by their Equation 1, is the worst-case loss within a ball, $\max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} L(\mathbf{w} + \boldsymbol{\epsilon})$. If one assumes that L is locally well-approximated by its quadratic Taylor approximation, and if one assumes that $\mathbf{w}$ is a local minimizer (so that the gradient vanishes), then this idealized sharpness metric is equivalent to $L(\mathbf{w}) + \frac{\rho^2}{2} \lambda_{\max}(\mathbf{w})$. In the more general situation where we make the quadratic Taylor approximation but $\mathbf{w}$ is not a local minimum, the idealized sharpness metric is upper-bounded by $L(\mathbf{w}) + \rho\|\nabla L(\mathbf{w})\| + \frac{\rho^2}{2}\lambda_{\max}(\mathbf{w})$ (see Appendix A). In practice, Foret et al. [1] do not propose to penalize their idealized metric; instead, they propose to penalize a proxy for $\rho\|\nabla L(\mathbf{w})\|$, the first-order version of their idealized metric.

The SAM algorithm has a hyperparameter ρ which controls the strength of the sharpness penalty. We use SAM[3] to train non-normalized VGG networks on CIFAR10 with cross-entropy loss until we achieve 99% train accuracy. We consider $\eta = 0.005$ (Figure 2) and $\eta = 0.05$ (Figure 3) in both the small-batch (red lines) and large-batch (blue lines) regime. At small batch sizes, SAM works as intended: in Figures 2 and 3, in red, we plot both the test accuracy (left column) and $\lambda_{\max}$ (right column) as we vary ρ, the SAM hyperparameter. Observe that for the small batch size, a higher ρ (higher sharpness penalty) causes the test accuracy to be higher, and $\lambda_{\max}$ to be lower.

However, at large batch sizes, SAM does not work as intended. Plotting the test accuracy and $\lambda_{\max}$, we find that for large batch size, higher ρ (sharpness penalty) still causes $\lambda_{\max}$ to shrink, but this no longer improves test accuracy. Indeed, at the large batch size, a higher ρ can lower the final test accuracy, even as it finds smaller $\lambda_{\max}$.

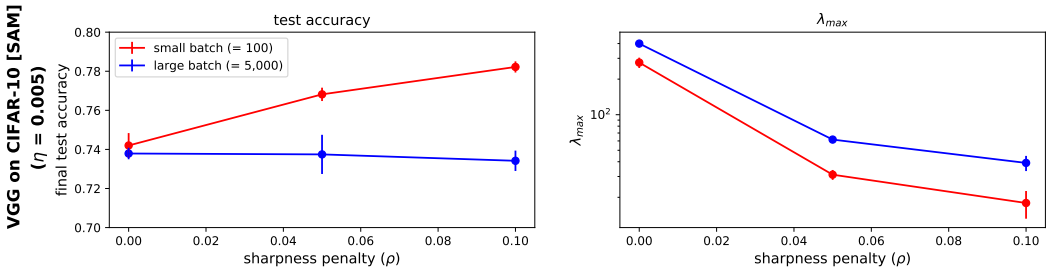

Figure 2: **SAM only exhibits generalization benefits for small batch sizes.** We train a VGG11 (no BN) using SAM until achieving 99% train accuracy at a range of values for ρ with a learning rate of 0.005. We plot the final test accuracy and $\lambda_{\max}$ (averaged over 4 runs with different random seeds) as a function of ρ. Observe that as ρ increases, the final test accuracy increases for small batch and decreases (slightly) for large batch. In both cases, as ρ increases, $\lambda_{\max}$ decreases.

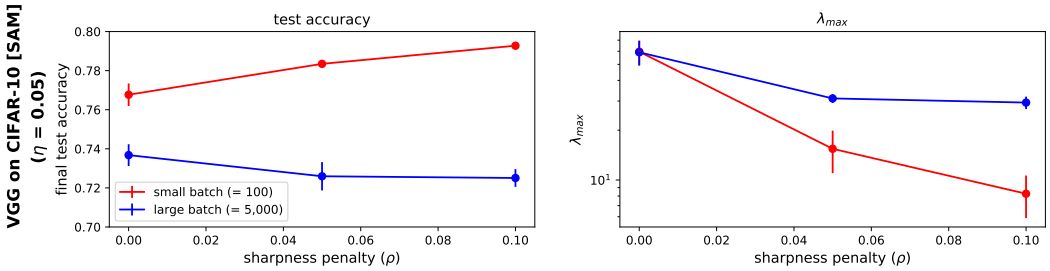

Figure 3: We use the same setup as in Figure 2, but with the large learning rate 0.05. The results are consistent with Figure 2.

3.3 Experiments on the Linear Scaling Rule, Dropout in DNNs, Batch Normalization in DNNs

We include the rest of our experiments in Appendix B. We show that by scaling batch size and learning rate simultaneously, we can change $\lambda_{\max}$ without affecting generalization (see Appendix B.2). We also show that excessively high dropout probabilities can degrade generalization, even as they promote smaller $\lambda_{\max}$ (see Appendix B.3). We then provide one example of an intervention that improves generalization, without changing $\lambda_{\max}$. Namely, we show that while batch-normalization does not consistently produce smaller $\lambda_{\max}$, it nevertheless confers generalization benefits (see Appendix B.4).

4 Conclusion

We have demonstrated that by manipulating the learning rate, batch size, training algorithm, and dropout, we can decrease the $\lambda_{\max}$ of a solution; yet these solutions with small $\lambda_{\max}$ do not always

[3] We use David Samuel's PyTorch implementation of SAM: https://github.com/davda54/sam.

generalize better. Similarly, we demonstrated that we can boost generalization performance without promoting small $\lambda_{\max}$, e.g., by using batch normalization. Our findings add to the growing body of work that calls into question the influence of flatness on generalization [13]. Since this paper refers to flatness as $\lambda_{\max}$ (the leading Hessian eigenvalue), note that our findings do not necessarily hold for other flatness metrics (e.g., the Hessian trace). While methods motivated by flatness produce useful tools [1], $\lambda_{\max}$ does not provide a scientific explanation for improvements in generalization. Thus, it is evident that there is a deeper story behind why flatness seems to be fruitful intuition. We hope to inspire future efforts aimed at understanding the relationship between $\lambda_{\max}$ and generalization, and to facilitate further discussion regarding whether $\lambda_{\max}$ should be treated as a generalization metric at all.

References

[1] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[2] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530, 2016. URL http://arxiv.org/abs/1611.03530.

[3] Leslie N. Smith. No more pesky learning rate guessing games. *CoRR*, abs/1506.01186, 2015. URL http://arxiv.org/abs/1506.01186.

[4] Samuel L. Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V. Le. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations (ICLR)*, 2018.

[5] Fengxiang He, Tongliang Liu, and Dacheng Tao. Control batch size and learning rate to generalize well: Theoretical and empirical evidence. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[6] Preetum Nakkiran. Learning rate annealing can provably help generalization, even for convex problems. *CoRR*, abs/2005.07360, 2020. URL https://arxiv.org/abs/2005.07360.

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

[8] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[9] Stanislaw Jastrzebski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd. In *Artificial Neural Networks and Machine Learning (ICANN)*, 2018.

[10] 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 *International Conference on Learning Representations (ICLR)*, 2017.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima, 1996.

[12] Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism, 2020.

[13] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets, 2017.

[14] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD:

training imagenet in 1 hour. *CoRR*, abs/1706.02677, 2017. URL http://arxiv.org/abs/1706.02677.

[15] Samuel L. Smith and Quoc V. Le. A bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations (ICLR)*, 2018.

[16] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*, 2019.

[17] Yeming Wen, Kevin Luk, Maxime Gazeau, Guodong Zhang, Harris Chan, and Jimmy Ba. An empirical study of large-batch stochastic gradient descent with structured covariance noise. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[18] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. In *International Conference on Learning Representations (ICLR)*, 2017.

[19] Xiangning Chen, Cho-Jui Hsieh, and Boqing Gong. When vision transformers outperform resnets without pretraining or strong data augmentations. *CoRR*, abs/2106.01548, 2021. URL https://arxiv.org/abs/2106.01548.

[20] Jean Kaddour, Linqing Liu, Ricardo Silva, and Matt J. Kusner. Questions for flat-minima optimization of modern neural networks. *CoRR*, abs/2202.00661, 2022. URL https://arxiv.org/abs/2202.00661.

[21] Diego Granziol. Flatness is a false friend, 2020.

[22] Shuofeng Zhang, Isaac Reid, Guillermo Valle Pérez, and Ard Louis. Why flatness does and does not correlate with generalization for deep neural networks, 2021. URL https://arxiv.org/abs/2103.06219.

[23] Stanislaw Jastrzebski, Maciej Szymczak, Stanislav Fort, Devansh Arpit, Jacek Tabor, Kyunghyun Cho, and Krzysztof Geras. The break-even point on optimization trajectories of deep neural networks. In *International Conference on Learning Representations (ICLR)*, 2020.

[24] 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. In *International Conference on Learning Representations (ICLR)*, 2021.

[25] Lei Wu, Chao Ma, and Weinan E. How sgd selects the global minima in over-parameterized learning: A dynamical stability perspective. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[26] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *CoRR*, abs/1404.5997, 2014. URL http://arxiv.org/abs/1404.5997.

[27] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning, 2016. URL https://arxiv.org/abs/1606.04838.

[28] Gintare Karolina Dziugaite, Alexandre Drouin, Brady Neal, Nitarshan Rajkumar, Ethan Caballero, Linbo Wang, Ioannis Mitliagkas, and Daniel M. Roy. In search of robust measures of generalization. *CoRR*, abs/2010.11924, 2020. URL https://arxiv.org/abs/2010.11924.

[29] Johan Bjorck, Carla P. Gomes, and Bart Selman. Understanding batch normalization. *CoRR*, abs/1806.02375, 2018. URL http://arxiv.org/abs/1806.02375.

A An Upper Bound on the original Sharpness Aware Minimization (SAM) Objective

Below, we will show that one way to penalize an upper bound on the original Sharpness Aware Minimization objective would be to penalize the leading Hessian eigenvalue, $\lambda_{\max}$.

Foret et al. [1] define the Sharpness Aware Minimization problem as follows:

$$\min_{\mathbf{w}} \max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} L(\mathbf{w}+\boldsymbol{\epsilon}) + \gamma \|\mathbf{w}\|_2^2 \tag{1}$$

where ρ and γ are hyperparameters. Using a second-order Taylor approximation, we can see that

$$L(\mathbf{w}+\boldsymbol{\epsilon}) \approx L(\mathbf{w}) + \nabla L(\mathbf{w})^\top \boldsymbol{\epsilon} + \frac{1}{2}\boldsymbol{\epsilon}^\top (\nabla^2 L(\mathbf{w}))\boldsymbol{\epsilon}$$

Given some $\boldsymbol{\epsilon}$ with $\|\boldsymbol{\epsilon}\| \leq \rho$, $\exists\ \hat{\boldsymbol{\epsilon}} = \frac{\boldsymbol{\epsilon}}{\|\boldsymbol{\epsilon}\|}$. Since $\hat{\boldsymbol{\epsilon}}^\top (\nabla^2 L(\mathbf{w}))\hat{\boldsymbol{\epsilon}} \leq \lambda_{\max}$, $\boldsymbol{\epsilon}^\top (\nabla^2 L(\mathbf{w}))\boldsymbol{\epsilon} \leq \rho^2 \lambda_{\max}$.

Thus, observe that the following upper bound on the original SAM objective explicitly regularizes $\lambda_{\max}$.

$$\begin{aligned}
\max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} L(\mathbf{w}+\boldsymbol{\epsilon}) &\approx \max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} L(\mathbf{w}) + \nabla L(\mathbf{w})^\top \boldsymbol{\epsilon} + \frac{1}{2}\boldsymbol{\epsilon}^\top (\nabla^2 L(\mathbf{w}))\boldsymbol{\epsilon} \\
&\leq L(\mathbf{w}) + \left[\max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} \nabla L(\mathbf{w})^\top \boldsymbol{\epsilon}\right] + \left[\max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} \frac{1}{2}\boldsymbol{\epsilon}^\top (\nabla^2 L(\mathbf{w}))\boldsymbol{\epsilon}\right] \\
&\leq L(\mathbf{w}) + \left[\max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} \|\nabla L(\mathbf{w})\| \|\boldsymbol{\epsilon}\|\right] + \left[\max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} \frac{1}{2}(\rho^2 \lambda_{\max})\right] \\
&\leq L(\mathbf{w}) + \rho \|\nabla L(\mathbf{w})\| + \frac{\rho^2}{2}\lambda_{\max}
\end{aligned}$$

In this sense, we can think of the hyperparameter ρ as the strength of the penalty imposed on sharpness.

Note that in practice, due to computational complexity, Foret et al. [1] use a first order Taylor approximation for the inner maximization problem and drop second order terms in the gradient approximation of the modified loss function.

B Experiments (continued)

B.1 Scaling learning rates in the small- and large-batch regimes (continued)

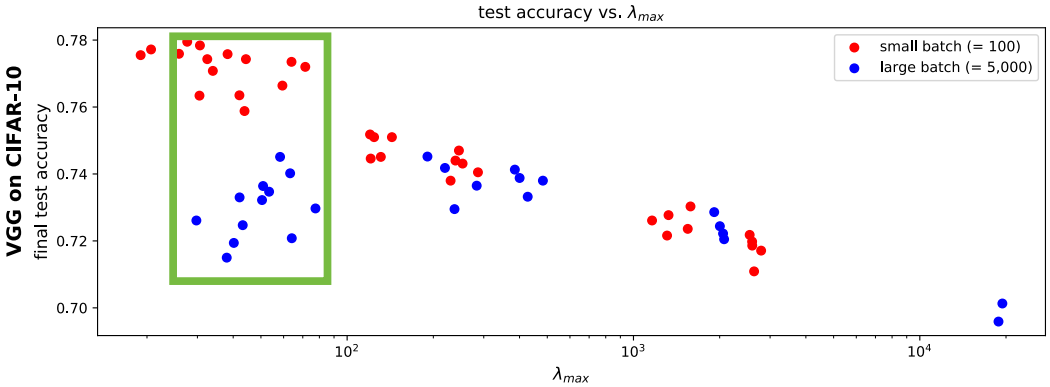

Figure 4: **Equivalent $\lambda_{\max}$ can generalize differently** We train a VGG11 (no BN) using SGD at a range of learning rates. We present a scatter plot of $\lambda_{\max}$ and test accuracy for both small and large-batch SGD. Observe that in the green box, there are multiple instances where $\lambda_{\max}$ of small-batch and large-batch training runs are similar, but the test accuracy is much better for small-batch SGD.

No absolute correlation between $\lambda_{\max}$ and test accuracy Observe that $\lambda_{\max}$ at small batch sizes is lower than $\lambda_{\max}$ at large-batch size (Figure 1). Thus, one might attempt to resuscitate the conventional wisdom by hypothesizing that a small $\lambda_{\max}$ is indeed correlated with good generalization, but that this effect only "kicks in" once $\lambda_{\max}$ is sufficiently low. However, this is not the case: observe that there are multiple regions in Figure 4 where $\lambda_{\max}$ of small and large-batch training runs are similar, but the test accuracy is much better for small-batch SGD. Thus, even ultra-small values of $\lambda_{\max}$ are not always associated with improved generalization.

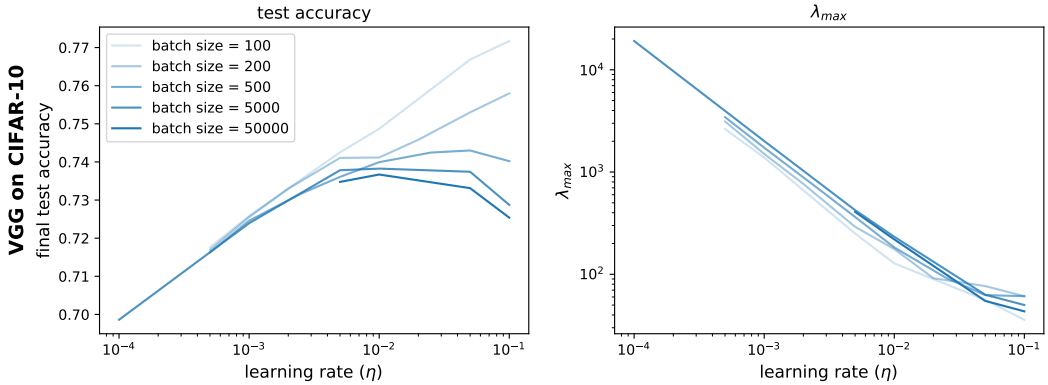

Figure 5: **Generalization benefits of large learning rates diminish (and sometimes reverse) as batch size increases** We show our findings from Figure 1 hold for a large range of batch sizes. Observe that as the batch size gets larger, the generalization benefits of large learning rates gradually diminish, yet large learning rates still find small $\lambda_{\max}$.

A gradual trend Previously, we presented data for only two batch sizes: 100 and 5000 (Figure 1). To fill in the picture, we present results for a range of batch sizes (Figure 5). Note that the effect we previously observed is gradual. As the batch size gets larger, the generalization benefits of large learning rates gradually diminish, yet large learning rates still find solutions with small $\lambda_{\max}$. As such, we show that λ_{max} is not a robust generalization measure in the case of large-batch/full-batch gradient descent, which is in line with the work in [28] on the success of a theory (in this case, of a generalization measure) for a family of environments.

B.2 When scaling learning rate and batch size simultaneously in the small-batch regime, $\lambda_{\max}$ and generalization aren't correlated

According to the linear scaling rule, in the small-batch regime, models trained with the same batch size to learning rate ratio (denoted by α) should exhibit similar generalization behavior [14, 26, 27, 9]. Meanwhile, prior work suggests that models trained with large learning rates should have small $\lambda_{\max}$ [9, 23]. Consequently, one might suspect that in the small-batch regime, if we scale learning rate and batch size simultaneously while keeping their ratio fixed, $\lambda_{\max}$ will get smaller while generalization will remain approximately unchanged—providing another counterexample in which generalization and $\lambda_{\max}$ are not correlated. We now confirm that this suspicion is true.

We confirm the linear scaling rule with respect to final test accuracy, but find that $\lambda_{\max}$ behaves differently (Figure 6). In particular, for a fixed α, observe that $\lambda_{\max}$ is large when the batch size (and corresponding learning rate) is small. We attribute this phenomenon to the influence of learning rate on $\lambda_{\max}$. In Figure 7, we plot the evolution of $\lambda_{\max}$ for a fixed α, using three different (batch size, η) pairs. We observe that larger learning rates lead to smaller $\lambda_{\max}$ throughout training. This is consistent with our observation in Figure 1, where larger learning rates lead to smaller $\lambda_{\max}$, regardless of the batch size.

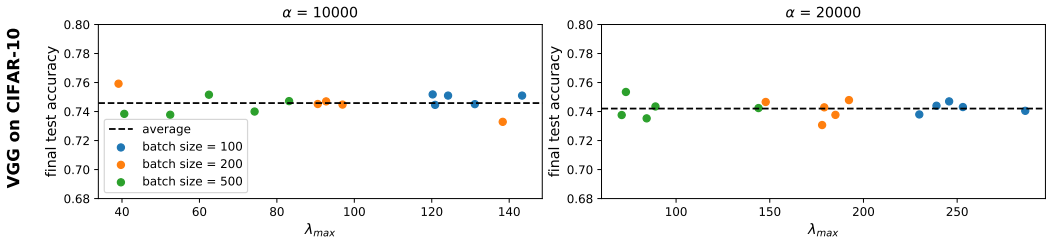

Figure 6: **Test accuracy, but not $\lambda_{\max}$, depends on ratio of batch size to learning rate** Let α denote the ratio of batch size to learning rate. We plot the final test accuracy vs. $\lambda_{\max}$ for various α, with 5 runs for every (batch size, η) pair. Observe that networks trained with the same α but different batch sizes exhibit similar final test accuracies, but different $\lambda_{\max}$.

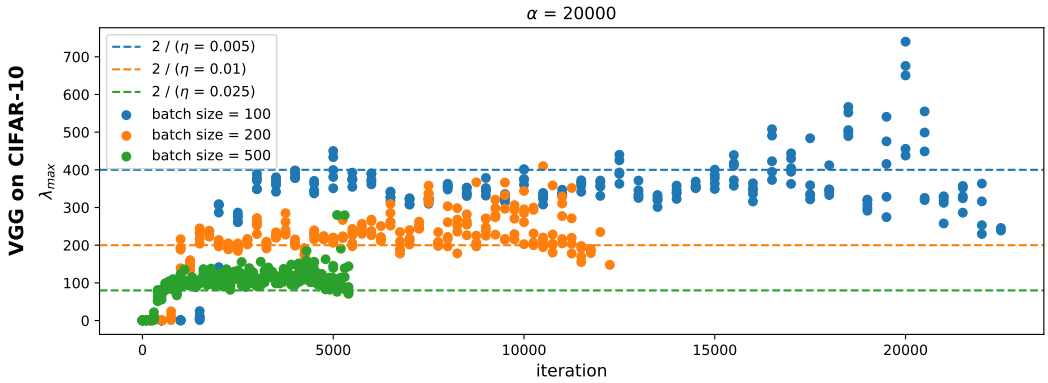

Figure 7: **For a fixed batch size to learning rate ratio, larger learning rates find smaller $\lambda_{\max}$** For $\alpha = 20000$, we plot the evolution of $\lambda_{\max}$ during training over 5 runs for every (batch size, η) pair. Observe that $\lambda_{\max}$ tends to remain the smallest for $\eta = 0.025$ and largest for $\eta = 0.005$.

B.3 Dropout in DNNs

We now demonstrate that while higher dropout [8] generally causes $\lambda_{\max}$ to shrink, but that at excessive dropout levels, these $\lambda_{\max}$ are accompanied by degradations in generalization (Figures 8 and 9). In these experiments, we train networks on Fashion-MNIST with cross-entropy loss and minibatch SGD until achieving 99% train accuracy. We consider various learning rates, batch sizes, and dropout probabilities p. In these experiments, our model is an MLP with 2 hidden layers, with dropout applied to both. Concretely, we make the following observations.

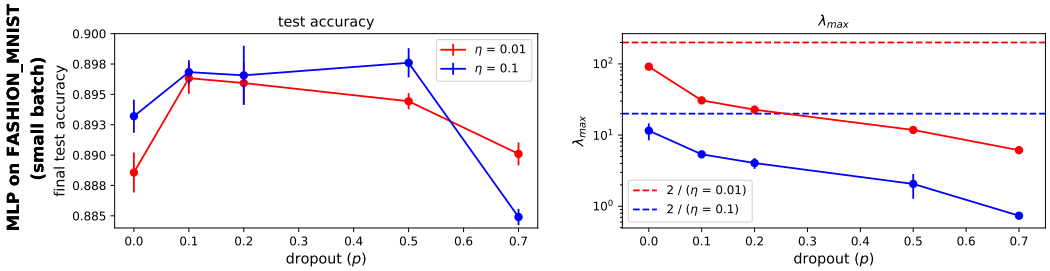

Figure 8: **Dropout induces small $\lambda_{\max}$, yet does not always boost generalization.** We train an MLP (with 2 hidden layers and dropout) using small-batch SGD at a range of learning rates. We plot the final test accuracy and $\lambda_{\max}$ (averaged over 3 runs with different random seeds) as a function of the dropout probability p. Observe that models with higher dropout probabilities (until $p = 0.2$) exhibit slight generalization benefits, and that higher values of p lead to lower $\lambda_{\max}$.

(i) For a fixed learning rate and batch size, modest dropout rates improve generalization We plot the test accuracy attained by the model with dropout probabilities p at varying learning rates

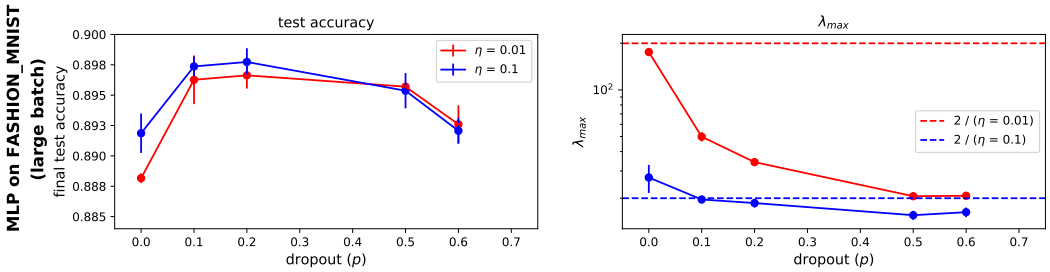

Figure 9: We use the same setup as in Figure 8 with a larger batch size. Our observations are consistent with those in Figure 8.

for both small-batch and large-batch (Figures 8 and 9). Observe that for both small-batch and large-batch SGD, adding dropout (up to $p = 0.2$) improves generalization.

(ii) High dropout probabilities promote small $\lambda_{\max}$ Higher dropout probabilities induce flatter solutions, as measured by smaller $\lambda_{\max}$ (Figures 8 and 9). This observation holds true in both the small-batch and large-batch regime. Note that we turn off dropout when computing $\lambda_{\max}$ because that is how predictions are generally made.

Recall that for a fixed learning rate, $\lambda_{\max}$ at small batch sizes is lower than $\lambda_{\max}$ at large batch sizes (see Figure 1 and [9, 23]). We hypothesize that randomness due to high dropout rates and stochasticity introduced by smaller batch sizes promote smaller $\lambda_{\max}$ by a similar mechanism.

(iii) However, excessively high dropout probabilities don't generalize better For each learning rate in both the small-batch and large-batch regimes, $\lambda_{\max}$ is lower for higher dropout rates. However, as the dropout probabilities are made *excessively* high, lower $\lambda_{\max}$ is accompanied by worse generalization. For example, for batch size 100 and $\eta = 0.1$, the solution found using $p = 0.7$ has a smaller $\lambda_{\max}$ than the one found using $p = 0.1$, yet generalizes worse (Figure 8).

B.4 Batch Normalization in DNNs

In each previous example, the interventions reduced $\lambda_{\max}$ but either did not change, or even degraded, generalization. We now demonstrate that adding Batch Normalization (BN) has the opposite character: it leaves $\lambda_{\max}$ unchanged, but alters generalization (Figures 10 and 11). In these experiments, we train VGG networks both with and without BN. As before, we train using minibatch SGD on CIFAR10 until achieving 99% train accuracy. We observe the following:

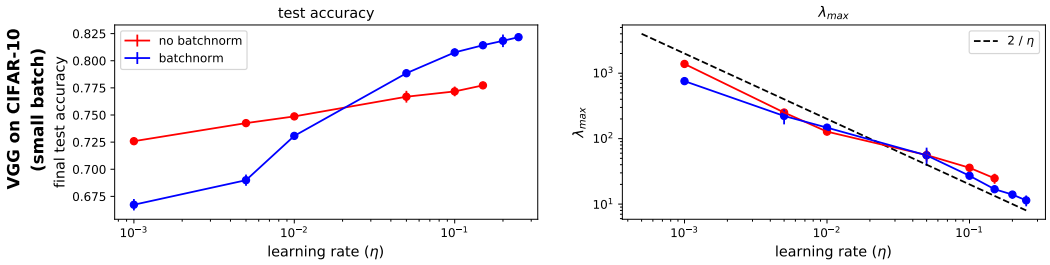

Figure 10: **BN can impact generalization without significantly affecting $\lambda_{\max}$** We train a VGG11 (+ BN) and VGG11 (no BN) using small-batch SGD at a range of learning rates. We plot the final test accuracy and $\lambda_{\max}$ (averaged over 4 runs with different random seeds) as a function of the learning rate. Observe that for a fixed learning rate, the VGG11 (+BN) and the VGG11 (no BN) exhibit different test accuracies, yet have comparably sharp solutions.

(i) Models with/without BN generalize differently We observe that for *small* learning rates, models with BN generalize *worse* than models without BN; but for *large* learning rates, models with

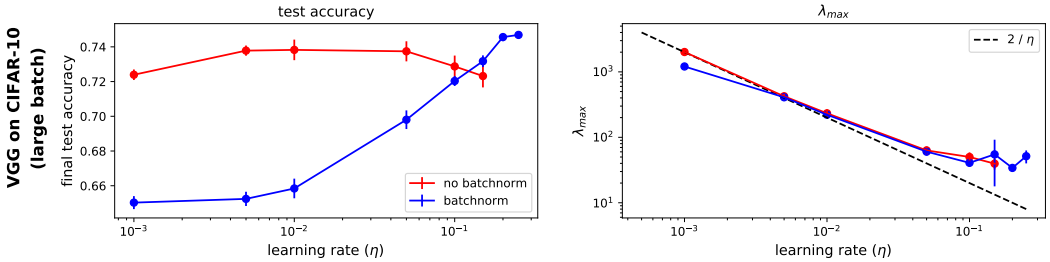

Figure 11: We use the same setup as in Figure 10 with a large batch size. Our observations are consistent with those in Figure 10.

BN generalize *better* than models without BN (Figures 10 and 11). Both findings are consistent with [29]. (While in Section 3.1, large learning rates did not improve generalization for unnormalized networks in the large-batch setting, we find here that for networks with BN enabled, large learning rates do improve generalization in the large-batch setting.)

We will now compare $\lambda_{\max}$ found by models with and without BN. As noted in Cohen et al. [24], for BN networks trained with small η, $\lambda_{\max}$ should be measured in between successive iterates, rather than directly at the iterates themselves. To estimate $\lambda_{\max}$ in this regime, we follow the procedure detailed in [24], where we compute the maximum $\lambda_{\max}$ over a grid of eight points spaced evenly between two successive iterates (specifically, we use the final iterate and train the network for one extra iteration).

(ii) However, for fixed learning rates and batch size, $\lambda_{\max}$ is not sensitive to BN We plot $\lambda_{\max}$ attained by SGD and BN at varying learning rates (Figures 10 and 11). Observe that in the large-batch setting (Figure 11), if we fix the learning rate, then $\lambda_{\max}$ found by models with and without BN are equivalent for most values of η. This is consistent with [24], which demonstrates that networks trained via full-batch gradient descent still train at the edge of stability in the presence of BN. Additionally, this finding provides more evidence in support of the idea that solutions with equivalent $\lambda_{\max}$ can generalize differently, as we observe in Figure 4. (Note that $\eta = 0.001$, $\lambda_{\max}$ found for the model with BN is slightly smaller than without BN. We suspect that $\lambda_{\max}$ does not match up exactly because we use a batch size of 5000, rather than the full batch; in general, smaller minibatches result in lower $\lambda_{\max}$ [23].) Likewise, in the small-batch setting (Figure 10), if we fix η, then $\lambda_{\max}$ found by models with and without BN are highly similar for most η, despite large differences in test accuracy. We acknowledge that in the small-batch setting, $\lambda_{\max}$ of un-normalized networks is slightly higher at large learning rates, and $\lambda_{\max}$ of normalized networks is lower at small learning rates. Importantly, in the latter case, normalized networks exhibit lower test accuracies, which is the *opposite* of what one might expect given the popular intuition that small $\lambda_{\max}$ cause good generalization.

C Additional Experiments

C.1 Linear Scaling Rule (continued)

We confirm that our findings from Section B.2 hold for a Fixup ResNet32 (see Figure 12).

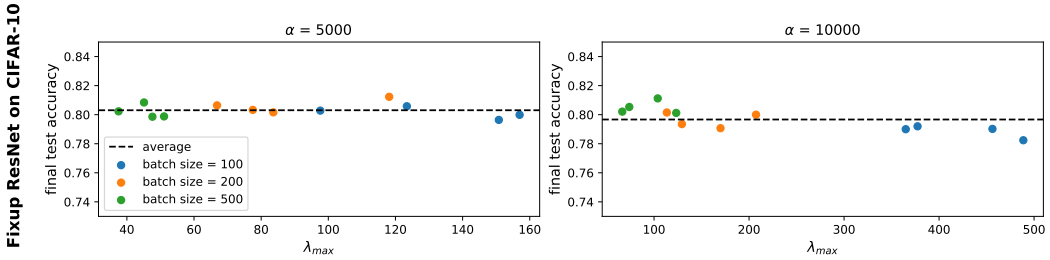

Figure 12: **Test accuracy, but not $\lambda_{\max}$, depends on ratio of batch size to learning rate** Let α denote the ratio of batch size to learning rate. We plot the final test accuracy vs. $\lambda_{\max}$ for various α, with 4 runs for every (batch size, η) pair. Observe that networks trained with the same α but different batch sizes exhibit similar final test accuracies, but different $\lambda_{\max}$. These findings are consistent with those in Figure 6.

C.2 SAM (continued)

For some architectures, large learning rates and SAM improve the final test accuracy in the large-batch regime, but for other architectures, they do not.

We confirm that for a Fixup ResNet32, SAM only exhibits generalization benefits for small batch sizes (see Figure 13). This is consistent with our findings in Section 3.2.

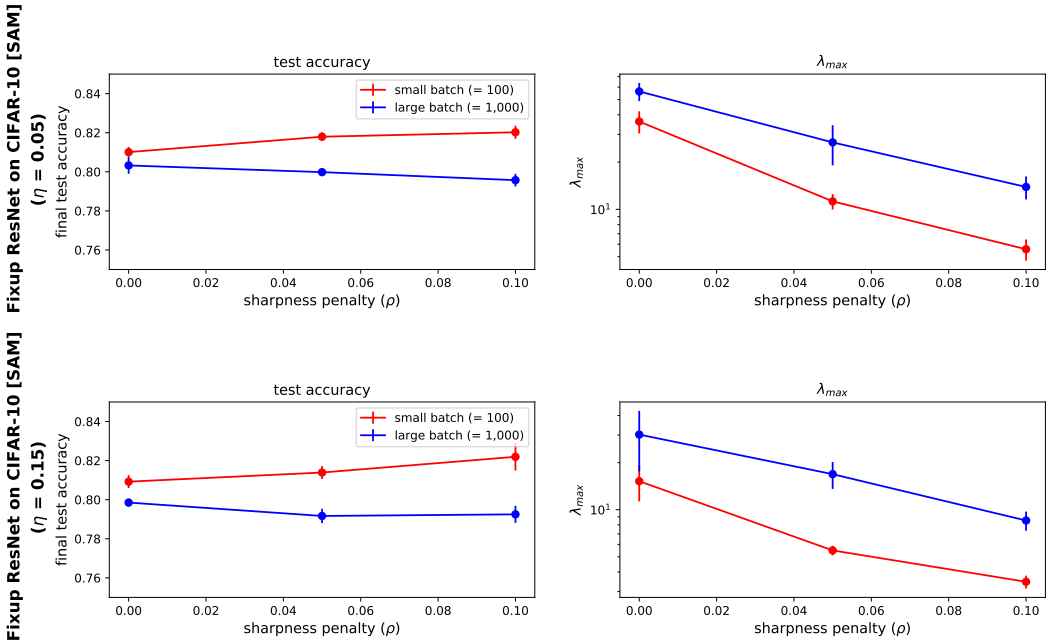

Figure 13: **For networks without BN, SAM only exhibits generalization benefits for small batch sizes.** We train a Fixup ResNet32 (no BN) using SAM until achieving 99% train accuracy at a range of values for ρ with learning rates of 0.05 and 0.15. We plot the final test accuracy and $\lambda_{\max}$ (averaged over 4 runs with different random seeds) as a function of ρ. Observe that as ρ increases, the final test accuracy increases for small batch and decreases (slightly) for large batch. In both cases, as ρ increases, $\lambda_{\max}$ decreases. These findings are consistent with those in Figures 2 and 3.

We note that large learning rates and SAM can improve final test accuracy in the large-batch regime in the case of BN networks (see Figure 14).

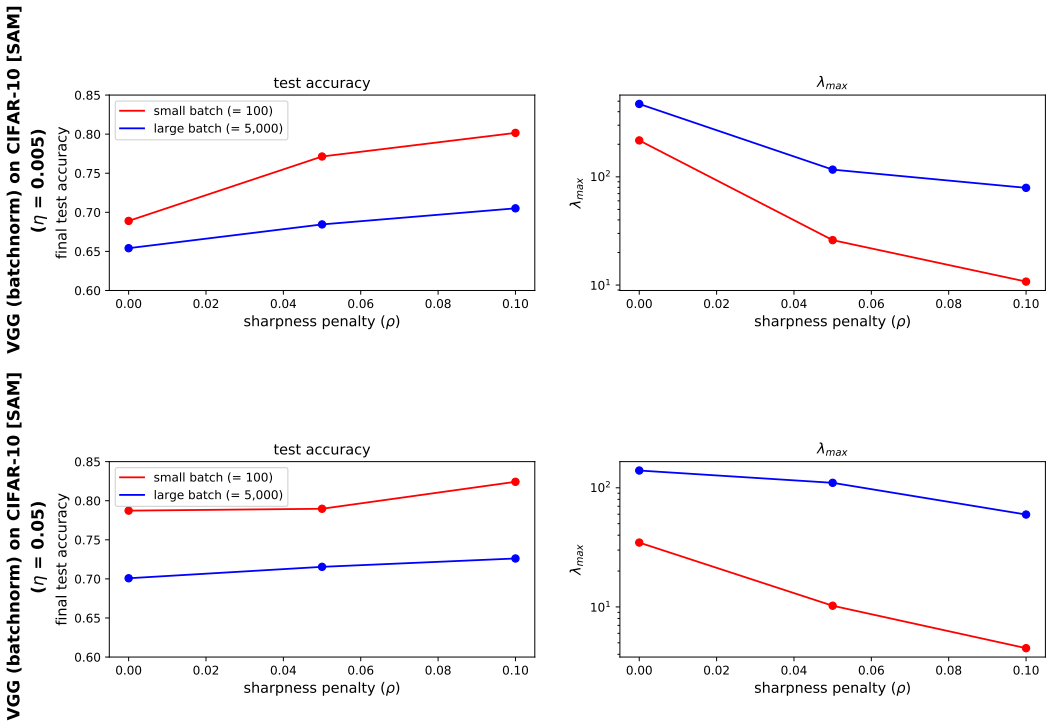

Figure 14: **SAM exhibits generalization benefits for small batch and large batch sizes for BN networks.** We train a VGG11 (with BN) using SAM until achieving 99% train accuracy at a range of values for ρ with learning rates of 0.005 and 0.05. We plot the final test accuracy and $\lambda_{\max}$ (over 1 run with a fixed seed) as a function of ρ. Observe that as ρ increases, the final test accuracy increases for both small batch and large batch. In both cases, as ρ increases, $\lambda_{\max}$ decreases.

C.3 Scaling learning rates in the small- and large-batch regimes (continued)

We run our experiments from Section 3.1 on a modified VGG11 (no BN), where we use GeLU in place of ReLU as the activation function. Observe that as the learning rate increases, the final test accuracy increases for small-batch and eventually decreases for large-batch SGD. In both cases, as the learning rate increases, $\lambda_{\max}$ decreases. These findings are consistent with those from Section 3.1.

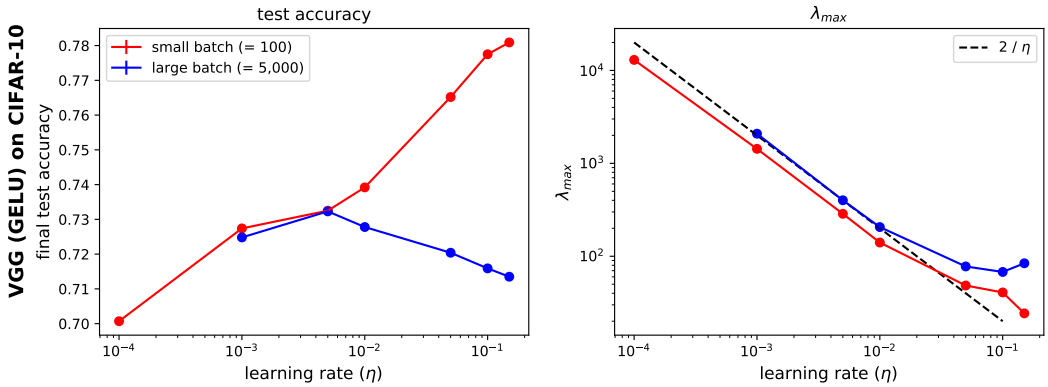

Figure 15: We use the same setup as in Figure 1, but with a modified VGG11 architecture, where all instances of ReLU as the activation function are replaced with GeLU. The results are consistent with Figure 1.