Journal of Machine Learning Research 23 (2022) 1-65 Submitted 11/20; Revised 5/22; Published 6/22

Learning Rates as a Function of Batch Size: A Random Matrix Theory Approach to Neural Network Training

Diego Granziol[*] DIEGO@PURESTRENGTH.AI
AI Theory Lab
Huawei
Gridiron building, 1 Pancras Square, Kings Cross, London, N1C 4AG

Stefan Zohren STEFAN.ZOHREN@ENG.OX.AC.UK
Stephen Roberts SJROB@ROBOTS.OX.AC.UK
Machine Learning Research Group and Oxford-Man Institute for Quantitative Finance
University of Oxford
25 Walton Well Rd, Oxford OX2 6ED, UK

Editor: Simon Lacoste-Julien

Abstract

We study the effect of mini-batching on the loss landscape of deep neural networks using spiked, field-dependent random matrix theory. We demonstrate that the magnitude of the extremal values of the batch Hessian are larger than those of the empirical Hessian. We also derive similar results for the Generalised Gauss-Newton matrix approximation of the Hessian. As a consequence of our theorems we derive an analytical expressions for the maximal learning rates as a function of batch size, informing practical training regimens for both stochastic gradient descent (linear scaling) and adaptive algorithms, such as Adam (square root scaling), for smooth, non-convex deep neural networks. Whilst the linear scaling for stochastic gradient descent has been derived under more restrictive conditions, which we generalise, the square root scaling rule for adaptive optimisers is, to our knowledge, completely novel. We validate our claims on the VGG/WideResNet architectures on the CIFAR-100 and ImageNet data sets. Based on our investigations of the sub-sampled Hessian we develop a stochastic Lanczos quadrature based on the fly learning rate and momentum learner, which avoids the need for expensive multiple evaluations for these key hyper-parameters and shows good preliminary results on the Pre-Residual Architecture for CIFAR-100. We further investigate the similarity between the Hessian spectrum of a multi-layer perceptron, trained on Gaussian mixture data, compared to that of deep neural networks trained on natural images. We find striking similarities, with both exhibiting rank degeneracy, a bulk spectrum and outliers to that spectrum. Furthermore, we show that ZCA whitening can remove such outliers early on in training before class separation occurs, but that outliers persist in later training.

Keywords: Deep Learning Theory, Random Matrix Theory, Loss Surfaces, Neural Network Training, Learning Rate Scaling, Adam, Adaptive Optimization, Square root rule

∗. corresponding author, currently at PureStrength Limited

1. Introduction

Deep Learning has taken computer vision and natural language processing tasks by storm. The observation that different critical points on the loss surface post similar test set performance has spawned an explosion of theoretical (Choromanska et al., 2015a,b; Pennington and Bahri, 2017) and empirical interest (Papyan, 2018; Ghorbani et al., 2019; Li et al., 2017; Sagun et al., 2016, 2017; Wu et al., 2017), in deep learning loss surfaces, typically through study of the eigenspectrum of the Hessian. Scalar metrics of the Hessian, such as the trace/spectral norm, have been related to generalisation (Keskar et al., 2016; Li et al., 2017). Under a Bayesian (MacKay, 2003) and minimum description length framework (Hochreiter and Schmidhuber, 1997), flatter minima generalise better than sharp minima. This has, however, been disputed recently (Dinh et al., 2017) due to a perceived lack of parameterisation invariance, with further work considering a parameterisation invariant flatness metric (Tsuzuku et al., 2020). Theoretical work on the Hessian of neural networks has shown that all local minima are close to the global minimum (Choromanska et al., 2015a) and that critical points of high index (i.e those with many negative eigenvalues) have high loss values (Pennington and Bahri, 2017). second-order optimisation methods (Bottou et al., 2018), use the Hessian (or positive semi definite approximations thereof, such as the Fisher information matrix). They more efficiently navigate along narrow and sharp valleys, making significantly more progress per iteration (Martens, 2010; Martens and Sutskever, 2012; Martens and Grosse, 2015; Dauphin et al., 2014) than first-order methods.

A crucial part of practical deep learning is the concept of sub-sampling or mini-batching. Instead of using the entire data set of size N to evaluate the loss, gradient or Hessian at each training iteration, only a small randomly chosen subset of size $B \ll N$ is used. This allows faster progress and lessens the computational burden tremendously. However, despite its widespread use in optimisation, the precise characterisation of the effects of mini-batching on the loss landscape and implications thereof, has not been thoroughly investigated. In this paper we show that:

- Under assumptions consistent with the optimisation paradigm, the fluctuations in the Hessian due to mini-batching can be modelled as a random matrix;
- For the feed forward, fully connected network with cross-entropy loss we expect the full Hessian to be low-rank and we provide extensive experiments along with a theoretical derivation to back up this assertion.
- When the eigenvalues of the full data set Hessian are well separated from the fluctuations matrix (which we define in Section 3.1) due to mini-batching, the extremal eigenvalues of the batch Hessian are given by the extremal eigenvalues of the full Hessian plus a term proportional to the ratio of the *Hessian variance* to the batch size. We verify this empirically for the VGG-16 network (Simonyan and Zisserman, 2014) on the CIFAR-100 data set;
- By a natural extension of our framework we can (and experimentally do) investigate the nature of the Hessian under the data generating distribution, which is a natural object when considering the true risk surface and generalisation;

- Our rigorous theoretical results predicts initial perfect scaling, diminishing returns and stagnation when increasing the batch size of stochastic gradient descent training (Golmant et al., 2018; Shallue et al., 2018). This result is crucial for understanding how to alter learning rate schedules when exploiting large batch training and data-parallelism, or when using limited GPU capacity for small or mobile devices. Whilst this result has been experimentally verified and derived previously (Goyal et al., 2017; Smith et al., 2017), the setting here is much more general and less restrictive than in previous work;

- As a consequence of our analysis of the batch Hessian, we provide a Lanczos algorithm based learning rate and momentum learner, which we show works effectively in training neural networks out of the box on a preliminary example.

- For adaptive-gradient methods where the damping parameter is fixed to a small value (such as the Adam default settings) we derive and verify the efficacy of a square root learning rate scaling with batch size. Specifically we mean that we expect a similar performance and training stability as we increase/decrease the learning rate with the square root of the batch size increase/decrease.

- We explicitly experimentally validate our proposed scaling rules, by scaling the largest learning rate which trains without divergence on the VGG-16 (Simonyan and Zisserman, 2014) architecture, for a batch size of 128, with no weight decay and batch normalisation. We show that alternative scaling rules break down and fail to train in the regime where they predict more aggressive scalings (larger learning rates) than our rules.

- We show that alternate scaling rules when they are more conservative, give sub-optimal validation errors and hence can be considered sub-optimal from a practical perspective. We relate this to the similarity of paths taken throughout the loss landscape. Where similar paths result in similar validation/test performance.

- We investigate the pervasiveness of the spectral outliers, bulk spectrum and apparent degeneracy in the spectra of neural network Hessians. Specifically, we find that outliers are only non-discernible in the case of training to a weight vector where no learning can be achieved by any subsequent learning rate drop. We further show that, whilst there are spectral outliers even at network initialisation, it is possible to remove such outliers using ZCA whitening. We find this is only possible at initialisation, with outliers returning later in training, as performance on the training set improves.

- We further show that the neural network Hessian spectrum associated with real data sets is visually similar to that of Multi-Layer Perceptrons operating on Gaussian Mixture Model data, opening up an exciting avenue of new (potentially analytically tractable) research.

The paper is structured as follows. The relevance of our work, key contributions and relationships to prior literature is detailed in Section 2. Section 6 Illustrates the main result for practitioners. Section 3 details the random matrix theory framework modelling the noise due to mini-batching – it states the assumptions, lemmas and proofs. Section 4 gives the theoretical main result. Section 5 extends the framework from Section 3 to

strictly positive-definite matrices such as the Generalised Gauss-Newton matrix, along with a theoretical and empirical investigation on the low rank approximation of the full data set Hessian in Section 7. Section 8 provides experimental validation for the theoretical claims. We discuss why we expect similar trajectories in weight space to give similar validation curves in Section 9. We then derive and verify as consequence of our framework a linear scaling rule for SGD in Section 10 along with a square root scaling rule for Adam in Section 11 as a function of batch size. We discuss the Hesssian under the data generating distribution in Section 12 and why for classification we always expect outliers in the spectra in Section 13.We further investigate the phase transition of the spectrum, showing that by using ZCA whitening we can bring spectra outliers into the bulk, but only at initialisation. Finally, we conclude in Section 14. Several appendices provide further details as referred to in the main text.

2. Motivation

For samples drawn independently from the training set, the stochastic gradient $\boldsymbol{g}_i(\boldsymbol{w}) \in \mathbb{R}^{P\times 1}$ in expectation is equal to the empirical gradient $\mathbb{E}(\boldsymbol{g}_i(\boldsymbol{w})) = \boldsymbol{g}(\boldsymbol{w})$ (Boyd and Vandenberghe, 2009; Nesterov, 2013). However, for the sample inverse Hessian $\boldsymbol{H}_i^{-1}(\boldsymbol{w}) \in \mathbb{R}^{P\times P}$, we note that $\mathbb{E}(\boldsymbol{H}_i^{-1}(\boldsymbol{w})) \neq \boldsymbol{H}^{-1}(\boldsymbol{w})$, as inversion is not a linear operation. By the spectral theorem, every Hermitian matrix, can be represented by its spectrum $\boldsymbol{H}(\boldsymbol{w}) = \sum_i^P \lambda_i \boldsymbol{\phi}_i \boldsymbol{\phi}_i^T$ and hence the spectrum of $\boldsymbol{H}_i(\boldsymbol{w})$ differs from that of $(1/N)\sum_{i=1}^N \boldsymbol{H}(\boldsymbol{w})$ or that of $\mathbb{E}(H(\boldsymbol{w}))$. Whilst this problem may at first seem intractable, under specific assumptions about the *matrix of fluctuations*, which characterises how the Hessian of a single sample varies from that of the full data set, we can evaluate this difference in spectrum analytically. In this paper we develop this idea with two different assumptions. We show that our theory well describes the perturbations between the batch and full data Hessians for large neural networks (VGG) with millions of parameters on regularly used data sets (CIFAR-100). We show that as consequences of our theorems, scaling rules as a function of batch size for both stochastic gradient descent and adaptive optimisers (which are different) follow naturally. We analyse the scaling rules, which are derived from our work, on other common networks and data sets, such as Residual networks (He et al., 2016) and ImageNet. We note that other concurrent analytical works on the Hessian have also used the VGG net as a reference network (Papyan, 2020).

2.1 Practical Applicability

How the loss surface changes as a function of mini-batch size, is of general interest to the greater problem of understanding deep learning. In particular, in the following we detail three practical applications which we identify.

Second-order optimisation: Mini-batching is prevalent in all (Martens and Grosse, 2015; Dauphin et al., 2014) deep learning second-order optimisation methods. Certain proofs of convergence for this class of methods explicitly require similarity between the spectra of the sub-sampled and full data set Hessians (Roosta-Khorasani and Mahoney, 2016). Hence, understanding the spectral perturbations due to mini-batching is important for some

theoretical results regarding second-order methods. We note, however, that alternative proof methods (Bollapragada et al., 2019; Moritz et al., 2016) don't require such assumptions.

Gradient-based optimisation: For gradient methods on convex functions, the convergence rate, optimal and maximal learning rates are functions of the Lipschitz constant (Nesterov, 2013), which is the infimum of the eigenvalues of the Hessian in the weight manifold. Hence understanding the largest eigenvalue perturbation due to mini-batching also has direct implications for their stability and convergence. Our framework prescribes a linear scaling rule up to a threshold for stochastic gradient descent. The works in Krizhevsky (2014); Goyal et al. (2017) also prescribe a linear scaling of the learning rate with batch size, however it is justified under the unrealistic assumption that the gradient is the same at all points in weight space. Jain et al. (2017) show linear parallelisation and then thresholding for least squares linear regression, assuming strong convexity. Our result holds for more general losses and does not assume strong convexity. Other work which considers the effect of batch sizes on learning rate choices and various optimisation algorithms, considers a constant as opposed to evolving Hessian and relies on assumptions of co-diagonalisability of the Hessian and covariance of the gradients (Zhang et al., 2019), which is not necessary in our framework.

Adaptive gradient optimisation: For adaptive or stochastic second-order methods using small damping and small learning rates, our theory prescribes a square root scaling procedure. Hoffer et al. (2017) also prescribe a square root scaling based on the co-variance of the gradients, for stochastic gradient descent (SGD) but not for adaptive methods. Our analysis expressly shows that the ways in which SGD and adaptive-gradient methods traverse the loss surface differ and this alters the optimal learning rate scaling as we increase the batch size. To the best of our knowledge no work has considered the difference in learning rate scalings between adaptive and non adaptive methods. In this work we expressly show (and empirically validate) that whilst for SGD we expect a linear learning rate scaling rule to hold as we increase/decrease the batch size up to a threshold, for Adam with small numerical stability constant (as is typical in practice) we expect a square root scaling rule.

2.2 Related Work

To the best of our knowledge no prior work has theoretically or empirically compared the Hessian of the full data set and that of a mini-batch and the consequences thereof.

Previous Loss Landscape Work: Previous works focusing on the loss landscape structure as a function of loss value (Choromanska et al., 2015a; Pennington and Bahri, 2017) assume normality and independence of the inputs and weights and often even more assumptions, such as i.i.d. Hessian elements and free addition (Pennington and Bahri, 2017) which means that we can simply add the spectra of two matrices. Removing these assumptions is considered a major open problem (Choromanska et al., 2015b), addressed in the deep linear case with squared loss (Kawaguchi, 2016). Furthermore, the Hessian spectra are not compatible with outliers, extensively observed in practice (Sagun et al., 2016, 2017; Ghorbani et al., 2019; Papyan, 2018). We address both concerns, by considering a field dependence structure (Götze et al., 2012), non-identical element variances and modelling the outliers explicitly as low-rank perturbations (Benaych-Georges and Nadakuditi, 2011). This may be of more general use to the community outside of our applications.

Similar Scaling Rules: Smith and Le (2017) derived optimal learning rate scalings, which were found to be linear by considering the scale of gradient noise and (assuming independent draws) the central limit theorem. This work was further extended (and experimentally verified) in Smith et al. (2017). This raises the question *Why should we consider the impact of curvature as opposed to gradient variance?* One simple pedagogical reason includes a holistic understanding in the limit of full-dataset training. In Smith and Le (2017) the noise scale is given by a factor $\frac{N-B}{NB}$, where N, B denote the data set size and batch size respectively. In the case where $N = B$, even when there is no noise, learning rate choices are dictated by the local curvature. This is already well known in the stochastic (convex and otherwise) optimisation literature (Rakhlin et al., 2011; Shamir and Zhang, 2013; Lacoste-Julien et al., 2012; Harvey et al., 2019), where proofs typically set a learning rate of $\frac{1}{\lambda t}$, where λ, t denotes the Lipshitz constant (which is an upper bound on the local Hessian maximum eigenvalue) and the iteration number respectively, showing the importance of considering curvature. As a consequence of this, we implement and present an online learning rate and momentum learner which uses the local sub-sampled curvature estimate to estimate appropriate values for these two coefficients. Another practical consideration, to the best of our knowledge novel in this paper, is the difference in learning rate scaling for adaptive methods compared to that of gradient descent. This forms a key contribution and motivation for our framework. Because our fine-grained analysis allows for an understanding of what happens to different regions of the spectrum when sub-sampling, we predict a new phenomenon unexplored in previous literature. Whilst Smith et al. (2017) argue that a linear scaling rate can also be used [1], we note from the corresponding figure in their text that, before the final sharp learning rate drop, the test accuracy for Adam diverges significantly as the learning rate drops and batch size increases. This implies that the linear scaling rate does not hold and hence warrants further investigation and in the authors opinion a novel framework. In this paper we show how a curvature based approach identifies that, for adaptive methods, a different regime holds compared to that of SGD. We experimentally validate this observation. As a further potential practical use case, which could form the basis of future work, our framework naturally extends to stochastic second-order optimisation methods (Nocedal and Wright, 2006) such as KFAC (Martens and Grosse, 2015). These approximate the eigenvalue/eigenvectors pairs of the batch Hessian. Hence, an understanding of how the eigenvalue/eigenvector estimations vary as a function of batch size becomes useful.

Hessian Analysis of DNNs: Papyan (2020) provides an extensive analysis of Deep Neural Network Hessians, developing an attribution strategy to various elements of the observed spectra which they empirically verify. Specifically this work builds upon that of Papyan (2018), which shows that the spectral outliers are attributable to the covariance of gradient class means and demonstrates that a mini-bulk, separated from the main bulk and outliers, is attributable to the cross-class gradient covariance and, further, that the main bulk is attributable to the within-class covariance. The paper demonstrates this experimentally by leveraging linear algebraic tools to plot the spectrum of $\log \boldsymbol{H}$ and by removing the components due to the within and cross class covariance from the spectrum. The paper also shows that increasing separation of the spectral outliers from the bulk distribution occurs with network depth and that. Furthermore they show for softmax regression on

1. Figure 4b page 5

a Gaussian mixture data set, that separation of the spectral outliers from the mini-bulk and separation of the mini-bulk from the bulk can be analytically related to generalistaion. The work also provides an alternative matrix to KFAC Martens and Grosse (2015) for second order optimisation called CFAC, which is shown to be a better approximation to the Generalised Gauss Newton matrix. Ghorbani et al. (2019) re-introduce the Lanczos (Meurant and Strakoš, 2006) algorithm to the machine learning community and validate its accuracy to double precision using only a limited number ($m = 90$) of Hessian vector products. They use this tool to investigate the Hessian spectral density on Imagenet and conclude that there remains significant negative spectral mass at the end of training and that the optimisation landscape seems to be smoother without residual connections. They also discuss spectral outlier suppression due to batch normalisation and argue that increasing the gradient contribution to flatter directions is inherently beneficial to the optimisation process. Whilst both of these works similarly focus on the Hessian and use similar tools to evaluate the spectrum, our principal focus is on how the sub-sampled batch Hessian deviates from the empirical (and or population) Hessian and the impacts this has on network training and hence the focus of the work, theoretical basis and approach are very different. Some of the ideas in this work are inspired by earlier unfinished work on the true loss surface (Granziol et al., 2018).

3. Random matrix theoretic approach to the Batch Hessian

For an input, output pair $[\boldsymbol{x}, \boldsymbol{y}] \in [\mathbb{R}^{d_x}, \mathbb{R}^{d_y}]$ and a given prediction function $h(\cdot;\cdot) : \mathbb{R}^{d_x} \times \mathbb{R}^P \rightarrow \mathbb{R}^{d_y}$, we consider the family of prediction functions parameterised by a weight vector $\boldsymbol{w}$, i.e., $\mathcal{H} := \{h(\cdot;\boldsymbol{w}) : \boldsymbol{w} \in \mathbb{R}^P\}$ with a given loss function $\ell(h(\boldsymbol{x};\boldsymbol{w}), \boldsymbol{y}) : \mathbb{R}^{d_y} \times \mathbb{R}^{d_y} \rightarrow \mathbb{R}$. In conjunction with statistical learning theory terminology, we denote the loss over our data generating distribution $\psi(\boldsymbol{x}, \boldsymbol{y})$, as the *true risk*.

$$R_{true}(\boldsymbol{w}) = \int \ell(h(\boldsymbol{x};\boldsymbol{w}), \boldsymbol{y}) d\psi(\boldsymbol{x}, \boldsymbol{y}), \tag{1}$$

with corresponding gradient $\boldsymbol{g}_{true}(\boldsymbol{w}) = \nabla R_{true}(\boldsymbol{w})$ and Hessian $\boldsymbol{H}_{true}(\boldsymbol{w}) = \nabla^2 R_{true}(\boldsymbol{w}) \in \mathbb{R}^{P\times P}$. Given a data set of size N, we only have access to the *empirical risk*

$$R_{emp}(\boldsymbol{w}) = \sum_{i=1}^{N} \frac{1}{N} \ell(h(\boldsymbol{x}_i;\boldsymbol{w}), \boldsymbol{y}_i), \tag{2}$$

empirical gradient $\boldsymbol{g}_{emp}(\boldsymbol{w}) = \nabla R_{emp}(\boldsymbol{w})$ and empirical Hessian $\boldsymbol{H}_{emp}(\boldsymbol{w}) = \nabla^2 R_{emp}(\boldsymbol{w})$. To further reduce computation cost, often only the batch risk

$$R_{batch}(\boldsymbol{w}) = \frac{1}{B} \sum_{i=1}^{B} \ell(h(\boldsymbol{x}_i;\boldsymbol{w}), \boldsymbol{y}_i), \tag{3}$$

(where $B \ll N$ is the number of batches) and the gradients $\boldsymbol{g}_{batch}(\boldsymbol{w})$, Hessians $\boldsymbol{H}_{batch}(\boldsymbol{w})$ thereof are accessed. The Hessian describes the curvature at that point in weight space $\boldsymbol{w}$ and hence the risk surface can be studied through the Hessian.

3.1 Properties of the fluctuation matrix

We write the stochastic batch Hessian as the deterministic empirical Hessian plus a perturbation due to the sampling noise[2]

$$\boldsymbol{H}_{batch}(\boldsymbol{w}) = \boldsymbol{H}_{emp}(\boldsymbol{w}) + \boldsymbol{\epsilon}(\boldsymbol{w}). \tag{4}$$

Notice that we may write,

$$\epsilon(\boldsymbol{w}) = \frac{1}{B}\sum_{i}^{B}\Big(\boldsymbol{H}_i(\boldsymbol{w}) - \boldsymbol{H}_{emp}(\boldsymbol{w})\Big) \tag{5}$$

which has expectation of 0, as $\mathbb{E}[\boldsymbol{H}_i(\boldsymbol{w})] = \boldsymbol{H}_{emp}(\boldsymbol{w})$. The variance, assuming that sampling is performed *without replacement* (as is typical for practical Deep Learning), is given by[3]

$$\mathrm{Var}[\boldsymbol{\epsilon}(\boldsymbol{w})] = \frac{1}{B^2}\Big(\sum_{i}^{B}\mathrm{Var}(\boldsymbol{H}_i(\boldsymbol{w}))+\sum_{i\neq j}^{B}\mathrm{Cov}(\boldsymbol{H}_i(\boldsymbol{w}),\boldsymbol{H}_j(\boldsymbol{w})\Big) = \left(\frac{1}{B}-\frac{B-1}{B(N-1)}\right)\mathrm{Var}[\boldsymbol{H}_i(\boldsymbol{w})] \tag{6}$$

Here, the pre-factor can be approximated as $\approx \frac{1}{B} - \frac{1}{N}$ if we consider $B > 1$ and $N \gg 1$, as is typically the case in Deep Learning. Note that this variance is strictly smaller when sampling without replacement than the corresponding variance obtained *with replacement*, which scales as $\frac{1}{B}$. Hence,

$$\mathbb{E}(\boldsymbol{\epsilon}(\boldsymbol{w})_{j,k}) = 0 \text{ and } \mathbb{E}(\boldsymbol{\epsilon}(\boldsymbol{w})_{j,k})^2 \approx \left(\frac{1}{B} - \frac{1}{N}\right)\mathrm{Var}[\nabla^2\ell(\boldsymbol{x},\boldsymbol{w};\boldsymbol{y})_{j,k}]. \tag{7}$$

where B is the batch size, N the total data set size and we use the fact that for each sample a given Hessian element has a common mean and variance. The expectation is taken with respect to $\psi(\boldsymbol{x},\boldsymbol{y})$. In order for the variance in Equation 7 to exist, the elements of $\nabla^2\ell(\boldsymbol{w},\boldsymbol{w};\boldsymbol{y})$ must obey sufficient moment conditions. This can either be assumed as a technical condition, or alternatively derived under the more familiar condition of L-Lipschitz continuity, as shown with the following Lemma

Lemma 1 *For a Lipschitz-continuous empirical risk gradient and almost everywhere twice differentiable loss function $\ell(h(\boldsymbol{x};\boldsymbol{w}),\boldsymbol{y})$, the elements of the fluctuation matrix $\boldsymbol{\epsilon}(\boldsymbol{w})_{j,k}$ are strictly bounded in the range $-\sqrt{P}L \leq \boldsymbol{\epsilon}(\boldsymbol{w})_{j,k} \leq \sqrt{P}L$. Where P is the number of model parameters and L is the smoothness constant.*

Proof As the gradient of the empirical risk is L Lipschitz continuous and the empirical risk is the sum over the samples, the gradient of the batch risk is also Lipschitz continuous. As the difference of two Lipschitz functions is also Lipschitz, by the fundamental theorem of calculus and the definition of Lipschitz continuity the largest eigenvalue λ_{max} of the

2. Note that although we could write $\boldsymbol{H}_{emp}(\boldsymbol{w}) = \boldsymbol{H}_{batch}(\boldsymbol{w}) - \boldsymbol{\epsilon}(\boldsymbol{w})$, this treatment is not symmetric as $\boldsymbol{H}_{batch}(\boldsymbol{w})$ is dependent on $\boldsymbol{\epsilon}(\boldsymbol{w})$, whereas $\boldsymbol{H}_{emp}(\boldsymbol{w})$ is not.
3. Note that if the batch size $B = N$, that $\epsilon(\boldsymbol{w}) = 0 \forall (\boldsymbol{w})$

fluctuation matrix $\boldsymbol{\epsilon}(\boldsymbol{w})$ must be smaller than L. Hence using the Frobenius norm we can upper bound the matrix elements of $\boldsymbol{\epsilon}(\boldsymbol{w})$

$$\begin{aligned}\mathrm{Tr}(\boldsymbol{\epsilon}(\boldsymbol{w})^2) &= \sum_{j,k=1}^{P} \boldsymbol{\epsilon}(\boldsymbol{w})_{j,k}^2 = \boldsymbol{\epsilon}(\boldsymbol{w})_{j=j',k=k'}^2 + \sum_{j\neq j',k\neq k'}^{P} \boldsymbol{\epsilon}(\boldsymbol{w})_{j,k}^2 = \sum_{i=1}^{P} \lambda_i^2 \\ \text{thus } \boldsymbol{\epsilon}(\boldsymbol{w})_{j=j',k=k'}^2 &\leq \sum_{i=1}^{P} \lambda_i^2 \leq PL^2 \;\text{ and }\; -\sqrt{P}L \leq \boldsymbol{\epsilon}(\boldsymbol{w})_{j=j',k=k'} \leq \sqrt{P}L.\end{aligned} \tag{8}$$

■

As the domain of the Hessian elements under the data generating distribution is bounded, the moments of Equation 7 are bounded and hence the variance exists. We can even go a step further with the following extra lemma.

Lemma 2 *For independent samples drawn from the data generating distribution and an L-Lipschitz loss ℓ the difference between the empirical Hessian and Batch Hessian converges element-wise to a zero mean, normal random variable with variance $\propto \frac{1}{B} - \frac{1}{N}$ for large B, N.*

Proof By Lemma 1, the Hessian elements are bounded, hence the moments are bounded and using independence of samples and the central limit theorem (Stein, 1972)

$$(\frac{1}{B} - \frac{1}{N})^{-1/2}[\nabla^2 R_{emp}(\boldsymbol{w}) - \nabla^2 R_{batch}(\boldsymbol{w})]_{jk} \underset{a.s}{\longrightarrow} \mathcal{N}(0, \sigma_{jk}^2) \tag{9}$$

■

3.2 The fluctuation matrix spectrum converges to the semi-circle law

To derive analytic results, we employ the Kolmogorov limit (Bun et al., 2017), where $P, B, N \to \infty$ but $P(\frac{1}{B} - \frac{1}{N}) = q > 0$.

We preserve the shape factor q to keep our results consistent with the theoretical and applied random matrix theory literature (Baik and Silverstein, 2006; Bun et al., 2016, 2017). Mathematically, the limit to infinity allows for the convergence of stochastic quantities into deterministic ones, for which we can derive exact expressions. We discuss finite size corrections, both experimentally and state the corresponding theoretical corrections, in Section 8.2.1. Note that for typical deep learning the number of parameters is in the millions or billions, and the data set size is also often in the tens or thousands or millions of examples. State of the art training also utilises batch sizes in the thousands (Goyal et al., 2017). We experimentally demonstrate in our experiments that whilst for smaller batch sizes e.g $B = 128$, stochasticity is important, we find that the mean predictions given by our framework are still accurate and useful.

By Lemma 1, we have $\mathbb{E}(\boldsymbol{\epsilon}(\boldsymbol{w})_{j,k}) = 0$ and $\mathbb{E}(\boldsymbol{\epsilon}(\boldsymbol{w})_{j,k}^2) = \sigma_{j,k}^2$. To further account for dependence beyond the symmetry of the fluctuation matrix elements, we introduce the σ-algebras

$$\mathfrak{F}^{(i,j)} := \sigma\{\boldsymbol{\epsilon}(\boldsymbol{w})_{kl} : 1 \leq k \leq l \leq P, (k,l) \neq (i,j)\}, \quad q \leq i \leq j \leq P \tag{10}$$

We can now state the following Theorem which is based on a general result from Götze et al. (2012):

Theorem 3 *Under the conditions of Lemmas 1 and 2, where* $\boldsymbol{\epsilon}(\boldsymbol{w}) \equiv \boldsymbol{H}_{batch}(\boldsymbol{w}) - \boldsymbol{H}_{emp}(\boldsymbol{w})$ *along with the following technical conditions:*

(i) $\frac{1}{P^2}\sum_{i,j=1}^{P}\mathbb{E}|\mathbb{E}(\boldsymbol{\epsilon}(\boldsymbol{w})_{i,j}^2|\mathfrak{F}^{i,j}) - \sigma_{i,j}^2| \to 0$,

(ii) $\frac{1}{P}\sum_{i=1}^{P}|\frac{1}{P}\sum_{j=1}^{P}\sigma_{i,j}^2 - \sigma_{\epsilon}^2| \to 0$

(iii) $\max_{1\leq i\leq P}\frac{1}{P}\sum_{j=1}^{P}\sigma_{i,j}^2 \leq C$

when $P \to \infty$*, the limiting spectral density* $p(\lambda)$ *of* $\boldsymbol{\epsilon}(\boldsymbol{w}) \in \mathbb{R}^{P\times P}$ *satisfies the semicircle law* $p(\lambda) = \frac{\sqrt{4\sigma_{\epsilon}^2-\lambda^2}}{2\pi\sigma_{\epsilon}^2}$*. Where* $\mathbb{E}(\boldsymbol{\epsilon}(\boldsymbol{w})_{i,j}^2|\mathfrak{F}^{i,j})$ *denotes the expectation conditioned on the sigma algebra, which is different to the unconditional expectation* $\mathbb{E}(\boldsymbol{\epsilon}(\boldsymbol{w})_{i,j}^2|\mathfrak{F}^{i,j}) \neq \mathbb{E}(\boldsymbol{\epsilon}(\boldsymbol{w})_{i,j}^2) = \sigma_{i,j}^2$.

We note that under the assumption of independence between all the elements of $\boldsymbol{\epsilon}(\boldsymbol{w})$ we would have obtained the same result, as long as conditions *ii*) and *iii*) were obeyed. So in simple words, condition 9*i*) merely states that the dependence between the elements cannot be too large. For example completely dependent elements have a second moment expectation that scales as P^2 and hence condition (i) cannot be satisfied. Condition (ii) merely states that there cannot be too much variation in the variances per element and condition (iii) that the variances are bounded. Note that $\boldsymbol{\epsilon}(\boldsymbol{w})$ is a function of the current iterate $\boldsymbol{w}$ and hence its spectrum depends on the Hessian at that point.

Proof Lindenberg's ratio is defined as $L_P(\tau) := \frac{1}{P^2}\sum_{i,j=1}^{P}\mathbb{E}|\boldsymbol{\epsilon}(\boldsymbol{w})_{i,j}|^2\mathbb{1}(|\boldsymbol{\epsilon}(\boldsymbol{w})_{i,j}| \geq \tau\sqrt{P})$. By Lemma 2, the tails of the normal distribution decay sufficiently rapidly such that $L_P(\tau) \to 0$ for any $\tau > 0$ in the $P \to \infty$ limit. Alternatively, using the Frobenius identity and Lipschitz continuity $\sum_{i,j=1}^{P}\mathbb{E}|\boldsymbol{\epsilon}(\boldsymbol{w})_{i,j}|^2\mathbb{1}(|\boldsymbol{\epsilon}(\boldsymbol{w})_{i,j}| \geq \tau\sqrt{P}) \leq \sum_{i,j}^{P}\boldsymbol{\epsilon}(\boldsymbol{w})_{i,j}^2 = \sum_{i}^{P}\lambda_i^2 \leq PL^2$, $L_P(\tau) \to 0$ for any $\tau > 0$. By Lemma 2 we also have $\mathbb{E}(\boldsymbol{\epsilon}(\boldsymbol{w})_{i,j}|\mathfrak{F}^{i,j}) = 0$. Hence along with conditions $(i), (ii), (iii)$ the matrix $\boldsymbol{\epsilon}(\boldsymbol{w})$ satisfies the conditions in Götze et al. (2012) and the and the limiting spectral density $p(\lambda)$ of $\boldsymbol{\epsilon}(\boldsymbol{w}) \in \mathbb{R}^{P\times P}$ converges to the semi-circle law $p(\lambda) = \frac{\sqrt{4\sigma_{\epsilon}^2-\lambda^2}}{2\pi\sigma_{\epsilon}^2}$ (Götze et al., 2012). Götze et al. (2012) use the condition $\frac{1}{P}\sum_{i=1}^{P}|\frac{1}{P}\sum_{j=1}^{P}\sigma_{i,j}^2 - 1| \to 0$, however this simply introduces a simple scaling factor, which is accounted for in condition *ii*) and the corresponding variance per element of the limiting semi-circle. ■

4. Main Result

Having shown that the limiting spectral density of the fluctuations matrix converges to the semi-circle, we are now in a position to present the main result of this paper.

Theorem 4 *Under the assumption that* $\boldsymbol{H}_{emp}$ *is of low-rank* $r \ll P$*, the extremal eigenvalues* $[\lambda_1', \lambda_P']$ *of the matrix sum* $\boldsymbol{H}_{batch}(\boldsymbol{w}) = \boldsymbol{H}_{emp}(\boldsymbol{w}) + \boldsymbol{\epsilon}(\boldsymbol{w})$*, where* $\lambda_1' \geq \lambda_2' ... \geq \lambda_P'$ *and* $\boldsymbol{\epsilon}(\boldsymbol{w})$

is defined in Section 3.1 and obeys the conditions set out in Theorem 3, are given by

$$\lambda_1' = \left\{ \begin{array}{ll} \lambda_1 + \frac{P}{\mathfrak{b}}\frac{\sigma_\epsilon^2}{\lambda_1}, & \text{if } \lambda_1 > \sqrt{\frac{P}{\mathfrak{b}}}\sigma_\epsilon \\ 2\sqrt{\frac{P}{\mathfrak{b}}}\sigma_\epsilon, & \text{otherwise} \end{array} \right\}, \lambda_P' = \left\{ \begin{array}{ll} \lambda_P + \frac{P}{\mathfrak{b}}\frac{\sigma_\epsilon^2}{\lambda_P}, & \text{if } \lambda_P < -\sqrt{\frac{P}{\mathfrak{b}}}\sigma_\epsilon \\ -2\sqrt{\frac{P}{\mathfrak{b}}}\sigma_\epsilon, & \text{otherwise} \end{array} \right\}. \quad (11)$$

where $[\lambda_1, \lambda_P]$ are the extremal eigenvalues of $\boldsymbol{H}_{emp}(\boldsymbol{w})$, $\mathfrak{b} = B/(1 - B/N)$ occurs due to the random sub-sampling and B is the batch-size.[4] *Recall that σ_ϵ is defined in Theorem 3, through the limiting spectral density $p(\lambda)$ of $\boldsymbol{\epsilon}(\boldsymbol{w})$. This result holds in the $P, B, N \to \infty$ limit, where $P/\mathfrak{b}$ remains finite.*

In order to prove Theorem 4 we utilise the following Lemma, which is taken from Benaych-Georges and Nadakuditi (2011) and for which we outline the proof in Appendix A.1 for completeness.

Lemma 5 *Denote by $[\lambda_1', \lambda_P']$ the extremal eigenvalues of the matrix sum $\boldsymbol{M} = \boldsymbol{A} + \boldsymbol{\epsilon}(\boldsymbol{w})/\sqrt{P}$, where $\boldsymbol{A} \in \mathbb{R}^{P\times P}$ is a matrix of finite rank r with extremal eigenvalues $[\lambda_1, \lambda_P]$ and $\boldsymbol{\epsilon}(\boldsymbol{w}) \in \mathbb{R}^{P\times P}$ with limiting spectral density $p(\lambda)$ satisfying the semicircle law $p(\lambda) = \frac{\sqrt{4\sigma_\epsilon^2 - \lambda^2}}{2\pi\sigma_\epsilon^2}$. Then we have*

$$\lambda_1' = \left\{ \begin{array}{ll} \lambda_1 + \frac{\sigma_\epsilon^2}{\lambda_1}, & \text{if } \lambda_1 > \sigma_\epsilon \\ 2\sigma_\epsilon, & \text{otherwise} \end{array} \right\}, \lambda_P' = \left\{ \begin{array}{ll} \lambda_P + \frac{\sigma_\epsilon^2}{\lambda_P}, & \text{if } \lambda_P < -\sigma_\epsilon \\ -2\sigma_\epsilon, & \text{otherwise} \end{array} \right\}. \quad (12)$$

We now proceed with the proof of Theorem 4:

Proof The variance per element is a function of the batch size B and the size of the empirical data set N, as given by Lemma 2. Furthermore, unravelling the dependence in P (which is simply the matrix dimension) due to the definition of the Wigner matrix (shown in Appendix A) leads to Theorem 4. ■

Comments on the Proof: Although for clarity we only focus on the extremal eigenvalues, the proof as shown in Appendix A holds for all outlier eigenvalues which are outside the spectrum of the fluctuation matrix. The assumption that either $\boldsymbol{H}_{emp}$ or $\boldsymbol{\epsilon}(\boldsymbol{w})$ are low-rank is necessary to use perturbation theory in the proof. This condition could be relaxed if a substantial part of the eigenspectrum of $\boldsymbol{H}_{emp}$ were considered to be mutually free with that of $\boldsymbol{\epsilon}(\boldsymbol{w})$ (Bun et al., 2017). In Section 7 we derive a bound on the rank of a feed-forward network, which we show to be small for large networks and provide extensive experimental evidence that the full Hessian is in fact low-rank. In the special case that $\boldsymbol{\epsilon}(\boldsymbol{w})_{i,j}$ are i.i.d. Gaussian, the fluctuation matrix is the Gaussian Orthogonal Ensemble, proposed as the spectral density of the Hessian by Choromanska et al. (2015a). In this case, Theorem 4 can be proved more succinctly, which we detail in full in the Appendix A.

Remark 6 *Note that whilst we have considered the framework in which the batch Hessian is considered a perturbation of the full data set (empirical) Hessian (via an additive perturbation),*

4. Note that the factor $\mathfrak{b} = B/(1 - B/N)$ has appeared before in (Jastrzębski et al., 2018; Jain et al., 2017).

we could have alternatively considered the batch Hessian to be the true Hessian (i.e the data set under the data generating distribution) plus an additive perturbation, i.e.

$$\boldsymbol{H}_{batch}(\boldsymbol{w}) = \boldsymbol{H}_{true}(\boldsymbol{w}) + \boldsymbol{\epsilon}(\boldsymbol{w}). \tag{13}$$

This might be considered appropriate if each sample is only seen once, or as is typical in deep learning, the extent of the augmentation, e.g. random flips, crops with zero padding, rotations, colour variations, additions of Gaussian noise, are so extensive that no identical (or sufficiently similar) samples are ever seen by the optimiser twice. Note that under this framework, we simply need to replace $\mathfrak{b} \to B$ in our framework and we simply replace the maximal eigenvalue λ_1 of the full data set Hessian with that of the Hessian of the data generating distribution. Since such an extension is natural under the typical neural network training framework utilising many augmentations (such as random flipping, cropping with zero padding, rotations, translations, or the addition of Gaussian noise) and can be readily derived from our framework. We investigate the nature of the true Hessian in Section 12. Here we show that the empirical Hessian does indeed closely resemble that of the true Hessian.

5. Extension to Fisher information and other positive-definite matrices

In the case of Logistic regression, which is simply a 0 hidden layer neural network with cross-entropy loss, by the diagonal dominance theorem (Cover and Thomas, 2012), the Hessian is semi-positive-definite and positive-definite with the use of $L2$ regularisation. Hence an underlying fluctuation matrix which contains negative eigenvalues is unsatisfactory and we extend our noise model to cover the positive semi definite case. Commonly used positive semi-definite approximations to the Hessian in deep learning (Martens, 2014) include the Generalised Gauss-Newton matrix (GGN) matrix (Martens, 2010; Martens and Sutskever, 2012) and the Fisher information matrix (Martens and Grosse, 2015; Pennington and Worah, 2018), both used extensively for optimisation and theoretical analysis. Hence to understand the effect of mini-batching on these practically relevant optimisers, we must also extend our framework. Below we introduce the Generalised Gauss-Newton matrix.

The Generalised Gauss-Newton matrix: For some common activations and loss functions typical in deep learning, such as the cross-entropy loss and sigmoid activation the Generalised Gauss-Newton matrix is equivalent to the Fisher information matrix (Pascanu and Bengio, 2013). The Hessian may be expressed in terms of the activation σ at the output of the final layer $f(\boldsymbol{w})$ using the chain rule as $\boldsymbol{H} = \nabla^2\sigma(f(\boldsymbol{w}))$ with corresponding (i,j)'th component:

$$\boldsymbol{H}(\boldsymbol{w})_{ij} = \sum_{k=0}^{d_y}\sum_{l=0}^{d_y} \frac{\partial^2\sigma(f(\boldsymbol{w}))}{\partial f_l(\boldsymbol{w})\partial f_k(\boldsymbol{w})}\frac{\partial f_l(\boldsymbol{w})}{\partial w_j}\frac{\partial f_k(\boldsymbol{w})}{\partial w_i} + \sum_{k=0}^{d_y}\frac{\partial\sigma(f(\boldsymbol{w}))}{\partial w_k}\frac{\partial^2 f_k(\boldsymbol{w})}{\partial w_j \partial w_i}. \tag{14}$$

The first term on the RHS of Equation 14 is known as the Generalised Gauss-Newton (GGN) matrix. The rank of a product is the minimum rank of its products so the raank of the GGN matrix is upper bounded by $B \times d_y$. Following Sagun et al. (2017) due to the convexity of the loss ℓ with respect to the output $f(\boldsymbol{w})$ we rewrite the GGN matrix per sample as

$$\sum_{k,l=0}^{d_y}\sqrt{\frac{\partial^2\sigma(f(\boldsymbol{w}))}{\partial f_l(\boldsymbol{w})\partial f_k(\boldsymbol{w})}}\frac{\partial f_l(\boldsymbol{w})}{\partial w_j} \times \sqrt{\frac{\partial^2\sigma(f(\boldsymbol{w}))}{\partial f_l(\boldsymbol{w})\partial f_k(\boldsymbol{w})}}\frac{\partial f_k(\boldsymbol{w})}{\partial w_i} = \boldsymbol{J}_*\boldsymbol{J}_*^T, \tag{15}$$

where we define $\boldsymbol{J}_*$ in order to retain a similarity for the GGN matrix in the case of the squared loss function (Pennington and Bahri, 2017), which has the form $\boldsymbol{G}(\boldsymbol{w}) = \boldsymbol{J}\boldsymbol{J}^T$. There are many potential candidate noise models due to the effect of mini-batching. Examples include the free multiplicative and information plus noise model (Bun et al., 2016; Hachem et al., 2013).

Let us simply consider a mini-batching model where the transformed Jacobian, J^*, is perturbed by additive noise. Specifically,

$$\boldsymbol{J}^*_{batch}(\boldsymbol{w}) = \boldsymbol{J}^*_{emp}(\boldsymbol{w}) + \boldsymbol{\epsilon}(\boldsymbol{w}). \tag{16}$$

Under this framework, as $\mathbb{E}[\boldsymbol{\epsilon}(\boldsymbol{w})] = 0$,

$$\mathbb{E}(\boldsymbol{J}_* + \boldsymbol{\epsilon})(\boldsymbol{J}_* + \boldsymbol{\epsilon})^T = \boldsymbol{J}_*\boldsymbol{J}_*^T + \mathbb{E}\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T. \tag{17}$$

Note that in this case $\boldsymbol{\epsilon}(\boldsymbol{w}) \in \mathbb{R}^{P\times(B\times d_y)}$ and hence $\boldsymbol{\epsilon}(\boldsymbol{w})\boldsymbol{\epsilon}(\boldsymbol{w})^T \in \mathbb{R}^{P\times P}$. Whilst it is known that, for i.i.d. Normal entries for $\boldsymbol{\epsilon}(\boldsymbol{w})$, the spectrum of $\mathbb{E}\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T$ converges to the Marchenko-Pastur distribution (Marčenko and Pastur, 1967), the conditions can similarly be relaxed to those stated in Theorem 4 (Adamczak, 2011; Gotze et al., 2015; O'Rourke et al., 2012). Hence, with this assumption and in line with the previous derivation, we consider the finite rank perturbation of the Marchenko-Pastur density and arrive at the following result. For completeness, we derive the non-unit-variance Stieltjes transform of the Marchenko-Pastur distribution in Appendix B.

Theorem 7 *The extremal eigenvalue λ'_1 of the matrix $\boldsymbol{G}_{batch}$, where $\boldsymbol{G}_{emp}$ has extremal eigenvalue λ_1, is given by*

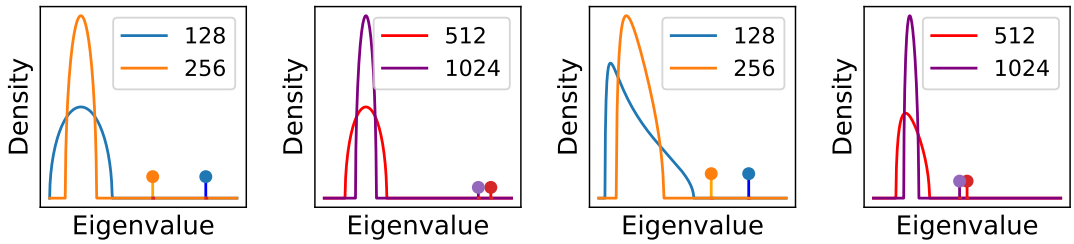
$$\lambda'_1 = \left\{ \begin{array}{ll} \frac{\lambda_1+\sigma^2(1-\frac{P}{\mathfrak{b}})}{1-\frac{P\sigma^2}{\lambda_1\mathfrak{b}}}, & \text{if } \lambda_1 > \sigma^2(1+\frac{P}{\mathfrak{b}}) \\ 2\sigma^2(1+\frac{P}{\mathfrak{b}}), & \text{otherwise} \end{array} \right\}. \tag{18}$$

The key conclusion is that, *independent of the exact limiting spectral density of the fluctuation matrix, we can consider the extremal eigenvalues of the True Hessian, or Generalised Gauss-Newton matrix (GGN), to be a low-rank perturbation of the fluctuation matrix. This can be considered a form of universality for the proved result in Theorem 4. Where the assumptions on the noise matrix may differ, but the key phenomena, that of spectra broadening, persists.*

How realistic is the low-rank approximation? Since this is a major assumption in our analysis, we investigate the experimental evidence for the low-rank nature of the empirical Hessian and empirical GGN in Section 7 and provide a theoretical argument for feed forward neural network Hessians.

6. Illustration of the Key Result

We illustrate our key result (formalised in Theorems 4 and 7 in Sections 3 & 5) in Figure 1. If the largest Hessian eigenvalue is well separated from the fluctuation matrix (continuous spectral density), as shown in Figures 1a & 1c, then increasing the batch size, which reduces the spectral width of the fluctuation matrix (which in turn reduces with the square root of the batch size), will have an approximately linear effect in reducing the spectral norm.



(a) Wigner Linear (b) Wigner Threshold (c) MP Linear (d) MP Threshold

Figure 1: **Variation of the spectral norm with batch size. Spectral norm decreases linearly until a threshold with batch size increase for both the Wigner and Machenko-Pastur noise models for mini-batching.** The continuous region (bulk) corresponds to the fluctuation matrix induced by mini-batching, shown as a Wigner semicircle (a & b) or Marchenko-Pastur (MP - c & d), whose width depends on the square root of the batch size). The largest eigenvalue of the batch Hessian is shown as a single peak (the largest outlier), which decreases in magnitude as the batch size increases.

This will hold up to a threshold, shown in Figures 1b & 1d, after which the spectral norm no longer appreciably changes in size. This is because the perturbation due to minibatch sampling no longer dominates the magnitude of the eigenvalue from the full data set Hessian. We discuss the prevalence and origin of spectral outliers in Deep Neural Network spectra in Section 13.

7. Evaluating the Low Rank Approximation

One of the key ingredients to proving Theorem 4, as shown in Section 3, is the use of perturbation theory. This requires either the fluctuation matrix, or the full empirical Hessian, to be low-rank. In our work, we consider the empirical Hessian to be low-rank. The rank degeneracy of small neural networks has already been discovered and discussed in Sagun et al. (2017) and reported for larger networks using spectral approximations in Ghorbani et al. (2019); Papyan (2018). We further provide extensive experimental validation for both the VGG-16 and PreResNet-110 on the CIFAR-100 data sets. However theoretical arguments rely on the Generalised Gauss-Newton matrix (GGN) decomposition. From Equation 14 it can be surmised that the rank of the GGN is bounded above by $N \times d_y$ (the data set size times the number of classes). However the Hessian is the sum of the GGN and another matrix, which has not been theoretically argued to be low-rank. The rank of a sum of two matrices is upper bounded by their rank sum. Furthermore, if the data set size becomes large (e.g. such as ImageNet with 10^7 entries) and the class number also large, even the GGN bound is ineffective. We hence provide in Section 7.1 a novel theoretical argument for a Hessian rank bound for feed-forward neural networks with a cross-entropy loss. The key intuition behind our proof is that each product of weights is a rank one object. Hence, if the sum of these products can be bounded we can also bound the rank. Since the sum depends on the number of neurons, the rank bound can end up becoming very small.

7.1 Theoretical argument for Feed Forward Networks

We consider a neural network with a d_x dimensional input $\boldsymbol{x}$. Our network has $H-1$ hidden layers and we refer to the output as the H'th layer and the input as the 0'th layer. We denote the ReLU activation function as $f(x)$ where $f(x) = \max(0, x)$. Let $\boldsymbol{W}_i$ be the matrix of weights between the $(i-1)$'th and i'th layer. For a d_y dimensional output our q'th component of the output can be written as

$$\boldsymbol{z}(\boldsymbol{x}_i; \boldsymbol{w})_q = f(\boldsymbol{W}_H^T f(\boldsymbol{W}_{H-1}^T ....f(\boldsymbol{W}_1 \boldsymbol{x}))) = \prod_{l=0}^{H} \sum_{n_{i,l}=1}^{N_l} \sum_{i}^{d_x} \boldsymbol{x}_i \boldsymbol{w}_{n_{i,l}, n_{i,l+1}} \tag{19}$$

where $\boldsymbol{w}_{n_{i,l}, n_{i,l+1}}$ denotes the weight of the path segment connecting node i in layer l with node i in layer $l+1$. layer l has N_l nodes. Where $n_{i,l_0} = x_i$. The Hessian, in the small loss limit tends to

$$\frac{\partial^2 \ell(h(\boldsymbol{x}_i; \boldsymbol{w}), \boldsymbol{y}_i)}{\partial w_{\phi,\kappa} \partial w_{\theta,\nu}} \to - \sum_{m \neq c} \exp(h_m) \left[\frac{\partial^2 h_m}{\partial w_{\phi,\kappa} \partial w_{\theta,\nu}} + \frac{\partial h_m}{\partial w_{\phi,\kappa}} \frac{\partial h_m}{\partial w_{\theta,\nu}} \right]. \tag{20}$$

$$\begin{aligned} \left[\frac{\partial^2 h_m}{\partial w_{\phi,\kappa} \partial w_{\theta,\nu}} + \frac{\partial h_m}{\partial w_{\phi,\kappa}} \frac{\partial h_m}{\partial w_{\theta,\nu}} \right] &= \prod_{l=1}^{d-1} \sum_{n_{i,l} \neq [(\phi,\kappa),(\theta,\nu)]}^{N_{i,l}} \sum_{i}^{d_x} \boldsymbol{x}_i \boldsymbol{w}_{n_{i,l}, n_{i,l+1}} \\ &+ \left(\prod_{l=1}^{d-1} \sum_{n_{i,l} \neq (\theta,\nu)}^{N_{i,l}} \sum_{i}^{d_x} \boldsymbol{x}_i \boldsymbol{w}_{n_{i,l}, n_{i,l+1}} \right) \left(\prod_{l=1}^{d-1} \sum_{n_{j,l} \neq (\phi,\kappa)}^{N_{j,l}} \sum_{i}^{d_x} \boldsymbol{x}_i \boldsymbol{w}_{n_{j,l}, n_{j,l+1}} \right) \end{aligned} \tag{21}$$

Each product of weights contributes an object of rank-1 (as shown in Section 2). Furthermore, the rank of a product is the minimum of the constituent ranks, i.e. $\text{rank}(AB) = \min \text{rank}(A, B)$. Hence Equation 21 is rank bounded by $2(\sum_l N_l + d_x)$, where N_l is the total number of neurons in the network. By rewriting the loss per-sample, repeating the same arguments and including the class factor, we obtain

$$\frac{\partial^2 \ell}{\partial w_k \partial w_l} = - \frac{\partial^2 h_{q(i)}}{\partial w_k \partial w_l} + \frac{\sum_j \exp(h_j) \sum_i \exp(h_i) (\frac{\partial^2 h_i}{\partial w_k \partial w_l} + \frac{\partial h_i}{\partial w_k} \frac{\partial h_i}{\partial w_l}) - \sum_i \exp(h_i) \frac{\partial h_i}{\partial w_k} \sum_j \frac{\partial h_j}{\partial w_l} \exp(h_j)}{[\sum_j \exp(h_j)]^2}, \tag{22}$$

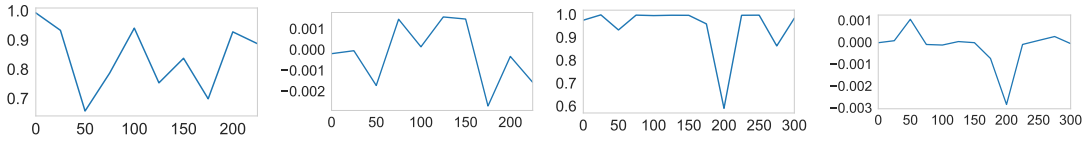
and thence a rank bound of $4d_y(\sum_l N_l + d_x)$. To give some context, along with a practical application of a real network and data set, for the CIFAR-10 data set, the VGG-16 (Simonyan and Zisserman, 2014) contains 1.6×10^7 parameters, the number of classes is 10 and the total number of neurons is $13,416$ and hence the bound gives us a spectral peak at the origin of at least $1 - \frac{577{,}600}{1.6 \times 10^7} = 0.9639$.

7.2 Experimental Validation of Low Rank Approximation

A full Hessian inversion with computational cost $\mathcal{O}(P^3)$ is infeasible for large neural networks. Hence, counting the number of zero eigenvalues (which sets the degeneracy) is not feasible in this manner. Furthermore, there would still be issues with numerical precision, so a threshold would be needed for accurate counting. Hence, based on our understanding of the Lanczos algorithm, discussed in Appendix D, we propose an alternative method.

Lanczos: We know that m steps of the Lanczos method, gives us an m-moment matched spectral approximation of the moments of $\boldsymbol{v}^T \boldsymbol{H} \boldsymbol{v}$, where in expectation over the set of zero mean, unit variance, random vectors this is equal to the spectral density of $\boldsymbol{H}$. Meurant and Strakoš (2006); Fitzsimons et al. (2017) Each eigenvalue/eigenvector pair estimated by the Lanczos algorithm is called a Ritz-value/Ritz-vector. We hence take $m \gg 1$, where for consistency with Ghorbani et al. (2019) we take $m = 100$ in our experiments[5]. We then take the Ritz value closest to the origin and take that as a proxy for the zero eigenvalue and report its weight.

Spectral Splitting: One weakness of this method is that for a large value of m, since the Lanczos algorithm finds a discrete moment matched spectral algorithm, the spectral mass near the origin may split into multiple components. Counting the largest thereof, or closest to the origin, may not be sufficient. We note this problem both for the PreResNet-110 and VGG-16 on the CIFAR-100 data set shown in Figure 2. Significant drops in degeneracy occur at various points in training and occur in tandem with significant changes in the absolute value of the Ritz value of minimal magnitude. This suggests the aforementioned splitting phenomenon is occurring. This issue is not present in the calculation of the Generalised Gauss-Newton matrix, as the spectrum is constrained to be positive-definite, so there is a limit to the extent of splitting that may occur. In order to remedy this problem for the Hessian, we calculate the combination of the two closest Ritz values around the centre and combine their mass. We consider this mass, and the weighted average of their values, as the degenerate mass. An alternative approach could be to kernel smooth the Ritz weights at their values, but this would involve another arbitrary hyper-parameter σ and hence we do not adopt this strategy.



(a) $\mathcal{D}$ weight P110 (b) $\mathcal{D}$ value P110 (c) $\mathcal{D}$ weight VGG-16 (d) $\mathcal{D}$ value VGG-16

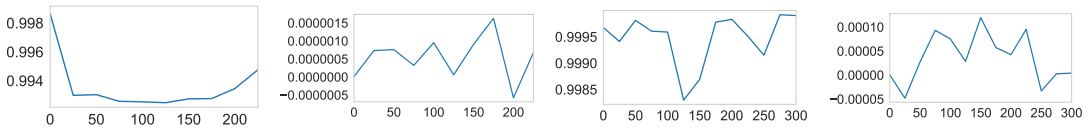
Figure 2: Rank degeneracy $\mathcal{D}$ (proportion of zero eigenvalues) evolution throughout training using the VGG-16 and PreResNet-110 on the CIFAR-100 data set, the weight corresponds to the spectral mass of the Ritz value(s) considered to correspond to $\mathcal{D}$

7.3 VGG16

For the VGG-16 model, which forms the reference model for this paper, we see that for both the Generalised Gauss-Newton matrix (GGN, shown in Figure 3a) and the Hessian (shown in Figure 3c) the rank degeneracy is extremely high. For the GGN, the magnitude of the Ritz value, which we take to be the origin, is extremely close to the threshold of GPU precision, as shown in Figure 3b. For the Hessian, for which we combine the two smallest absolute value Ritz values, we find an even larger spectral degeneracy. The weighted average also gives a value very close to 0, as shown in Figure 3d. The combined weighted average,

5. They show that $m = 90$ is sufficient for double precision accuracy on an MLP MNIST example

however, is much closer to the origin than that of the lone spectral peak, shown in Figure 2, which indicates splitting, we do not get as close to the GPU precision threshold of 10^{-7}, which we consider as a reasonable level to assume domination by numerical imprecision.

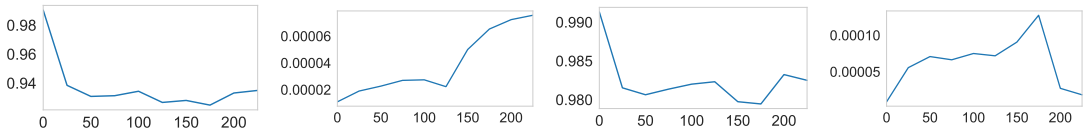


(a) GGN Degeneracy (b) GGN Ritz Value (c) Hessian Degeneracy (d) Hessian Ritz Value

Figure 3: Rank degeneracy (proportion of zero eigenvalues) evolution throughout training using the VGG-16 on the CIFAR-100 data set, total training 225 epochs, the Ritz value corresponds to the value of the node which we assign to 0.

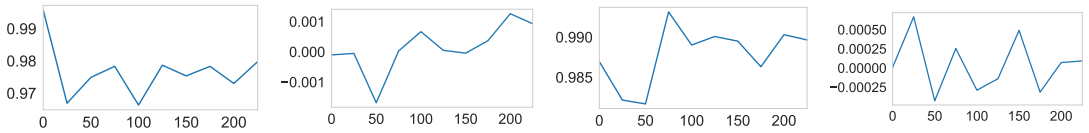
7.4 PreResNet110

We repeat the same experiments in Section 7.3 for the preactivated residual network with 110 layers, on the same data set. Note that, as explained in Section E, we can calculate the spectra in both batch normalisation and evaluation mode. Hence we report results for both, with the main finding that the empirical Hessian spectra are consistent with large rank degeneracy.



(a) BN-train Degen (b) BN-train Ritz Val (c) BN-eval Degen (d) BN-eval Ritz Val

Figure 4: Generalised Gauss-Newton matrix rank degeneracy (proportion of zero eigenvalues) evolution throughout training using the PreResNet-110 on the CIFAR-100 data set, total training 225 epochs, the Ritz value corresponds to the value of the node which we assign to 0.



(a) BN-train Degen (b) BN-train Ritz Val (c) BN-eval Degen (d) BN-eval Ritz Val

Figure 5: Hessian rank degeneracy (proportion of zero eigenvalues) evolution throughout training using the PreResNet-110 on the CIFAR-100 data set, total training 225 epochs, the Ritz value corresponds to the value of the node which we assign to 0.

8. Experimental Validation of the Theoretical Results

In this section we run experiments to explictly test the validity of our derived theorems, for which we then develop practical algorithms and scaling rules in the coming sections.

Experimental Setup: We use the GPU powered Lanczos quadrature algorithm (Gardner et al., 2018; Meurant and Strakoš, 2006), with the Pearlmutter trick (Pearlmutter, 1994) for Hessian and GGN vector products, using the PyTorch (Paszke et al., 2017) implementation of both Stochastic Lanczos Quadrature and the Pearlmutter. We then train a 16 Layer VGG CNN (Simonyan and Zisserman, 2014) with $P = 15291300$ parameters on the CIFAR-100 data set (45,000 training samples and 5,000 validation samples) using SGD and K-FAC optimisers. For both SGD and K-FAC, we use the following learning rate schedule:

$$\alpha_t = \begin{cases} \alpha_0, & \text{if } \frac{t}{T} \leq 0.5 \\ \alpha_0[1 - \frac{(1-r)(\frac{t}{T}-0.5)}{0.4}] & \text{if } 0.5 < \frac{t}{T} \leq 0.9 \\ \alpha_0 r, & \text{otherwise.} \end{cases} \tag{23}$$

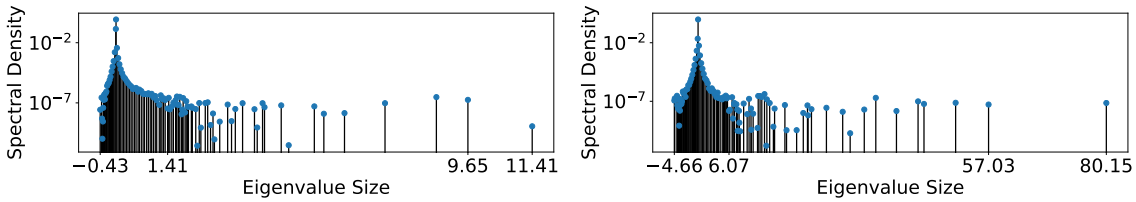
We use a learning rate ratio $r = 0.01$ and a total number of epochs budgeted $T = 300$. The number of training epochs is chosen largely for consistency with the results of previous work (Izmailov et al., 2018; Granziol et al., 2020) and the learning rate ratio similarly follows typical schedules of residual networks of two learning rate drops of factor 10 (He et al., 2016). We further use momentum set to $\rho = 0.9$ (also typical in residual networks (He et al., 2016)), a weight decay coefficient of 0.0005 (which we find gives the best performance for small data sets) and data-augmentation on PyTorch (Paszke et al., 2017). We set the inversion frequency to be once per 100 iterations for K-FAC.

Advantages of the VGG architecture: For simplicity, we do not analyse the added dependence between curvature and the samples due to batch normalisation (Ioffe and Szegedy, 2015) and hence adopt as our reference model the VGG-16 (Simonyan and Zisserman, 2014) on the CIFAR-100 data set which does not utilise batch normalisation. We show in Appendix E that many of our results also hold with batch-normalisation for ResNet architectures. We also include further results for the WideResNet architecture and the ImageNet-32 data set.

Estimating the Spectrum and Extremal Eigenvalues using the Lanczos Algorithm: To plot the spectrum of the neural network we use the approach of Granziol et al. (2019), which gives a discrete, moment-matched approximation to the underlying spectrum. We use $m = 100$ as the number of moments. As discussed in Granziol et al. (2019) the spectrum can be estimated consistently even using a single random vector, due to the high dimensionality of the neural network (large number of parameters). Whilst accurate bounds on the moments of the spectrum can be derived using stochastic Lanczos quadrature (Ubaru et al., 2017), we note that these bounds are considered very loose and pessimistic (Fitzsimons et al., 2017; Granziol and Roberts, 2017). Whether a spectrum, or more generally a density, can be accurately estimated using its moments is known as the Hausdorff moment problem (Hausdorff, 1921). It can be shown (Granziol and Roberts, 2017) that finite matrices (such as the Hessians of Neural Networks) satisfy these conditions. Hence there can be no surprises from "bad pathological spectra" in this case. Note that in the case of infinite matrices, we would need to have bounded moment conditions and hence finite eigenvalues, but this is not relevant for our measurements here.

8.1 Effect of spectral broadening for a typical batch size

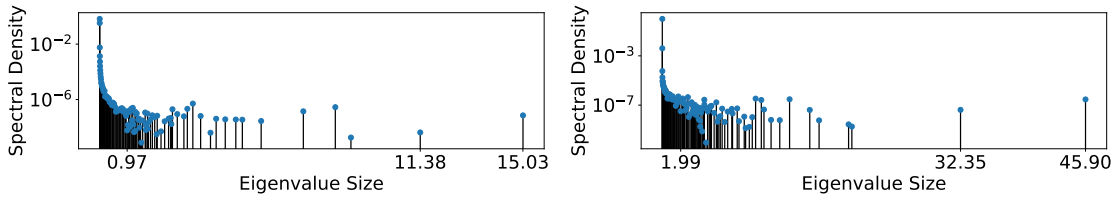
We plot an example effect of the spectral broadening of the Hessian due to mini-batching, for a typical batch size of $B = 128$ in Figure 6. *The magnitude of the extremal eigenvalues are significantly increased as are other outlier eigenvalues, such as the second largest.* We estimate the mean of the continuous region (bulk) of the spectrum as the position where the Ritz[6] weight drops below $1/P$. We see that the spectral width of this continuous region also increases. We plot an example of the Generalised Gauss-Newton matrix in Figure 7,



(a) Empirical Hessian spectrum $N = 50,000$ (b) Batch Hessian spectrum $B = 128$

Figure 6: Spectral Density of the Hessian at epoch 200, for different sample sizes B, N on a VGG-16 on the CIFAR-100 data set. The Y-axis corresponds to $p(\lambda)$ and the X-axis to λ. The initial learning rate used is $\alpha = 0.05$, with momentum $\rho = 0.9$ and weight decay 0.0005, using the learning rate schedule in Section 8.

which for cross-entropy loss and softmax activation is equal to the Fisher information matrix (Pascanu and Bengio, 2013). We observe identical behaviour of bulk and outlier broadening.



(a) Empirical GGN spectrum $N = 50,000$ (b) Batch GGN spectrum $B = 128$

Figure 7: Spectral Density of the Generalised Gauss-Newton matrix (GGN) at epoch 25 ,for different sample sizes B, N, on a VGG-16 on the CIFAR-100 data set. The Y-axis corresponds to $p(\lambda)$ and the X-axis to λ. The initial learning rate used is $\alpha = 0.05$, with momentum $\rho = 0.9$ and weight decay 0.0005, using the learning rate schedule in Section 8.

8.2 Measuring the Hessian Variance

We estimate the variance of the Hessian/GGN using stochastic trace estimation (Hutchinson, 1990; Granziol and Roberts, 2017) in Algorithm 1, from which the variance per element can be inferred. Note that under the assumptions of our model, which assumes that the batch Hessian is either a deterministic full Hessian plus a stochastic fluctuations matrix (Theorem

6. This is the term used by approximate eigenvalue/eigenvector pairs by the Lanczos algorithm, as detailed in Appendix D.

4), or alternatively the product of a stochastic fluctuations matrix and a deterministic modified Jacobian (Theorem 7), the variance of the elements of the Hessian directly leads us to the variance of the elements of the fluctuations matrix. We plot the evolution of the Hessian/GGN variance throughout an SGD training cycle in Figure 8, where we observe a slow initial growth, followed by explosive growth during learning rate reduction (from epoch 161 onwards) and then reduction when the learning rate is held fixed at a low value (from epoch 270 onwards). Because the variance of the Hessian massively increases in the later part of training (from epoch 161 onwards) and the variance of the Hessian determines the variance of the elements of the fluctuations matrix (because the full Hessian is deterministic). This Figure implies that we expect the batch Hessian extremal eigenvalues to diverge from those of the empirical Hessian during training. By 'diverge' we specifically mean substantially larger in magnitude. This is exactly what we see in practice in Figures 9c and 9d for both the Hessian and the Generalised Gauss Newton. Here we plot the batch Hessian maximum eigenvalues (Batch Maxval) using a batch size of $B = 128$ against the full Hessian maximum eigenvalues (Full MaxVal) over the course of training a VGG-16 on CIFAR-100. We track the Hessian variance over the trajectory to make our predictions (shown as Pert Maxval). We calculate the perturbation prediction using Theorems 4 and 7, where σ_ϵ is calculated using Algorithm 1. The full Hessian maximum eigenvalue used for the theorems and plotted is derived from using the Lanczos algorithm on the full data set N. We take the average of 10, $B = 128$ batch Hessian extremal eigenvalues. We shade in the $\pm$ standard deviation of our stochastic Batch Hessian and Batch Generalised Gauss Newton eigenvalues. We also repeat the same experiment for the KFAC optimiser and show similar results, pertaining to the difference between the full and batch eigenvalues along with the ability to predict them in Figures 9a, 9b. Whilst the goal of this section is to show that Theorems 4,7 are accurate and representative, we note that potentially accurate and cheap estimates of the full Hessian spectral norm could be calculated using an inverse procedure, whereby we calculate the spectral norm on a data subset and then, considering the Hessian variance within a subset, estimate the full Hessian spectral norm.

Algorithm 1 Calculate Hessian Variance

Input: Sample Hessian $\boldsymbol{H}_i \in \mathbb{R}^{P\times P}$
Output: Hessian Variance σ^2
$\boldsymbol{v} \in \mathbb{R}^{1\times P} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
Initialise $\sigma^2 = 0, i = 0$, $\boldsymbol{v} \leftarrow \boldsymbol{v}/||\boldsymbol{v}||$
for $i < N$ **do**
    $\sigma^2 \leftarrow \sigma^2 + \boldsymbol{v}^T \boldsymbol{H}_i^2 \boldsymbol{v}$
    $i \leftarrow i + 1$
end for
$\sigma^2 \leftarrow \sigma^2 - [\boldsymbol{v}^T (1/N \sum_{j=1}^{N} \boldsymbol{H}_j)\boldsymbol{v}]^2$

Table 1: Algorithm which estimates the central quantity $\sigma_{\boldsymbol{\epsilon}}^2$ in Theorems 4 & 7.

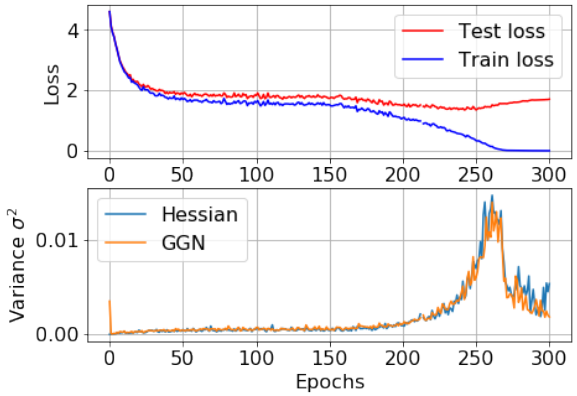
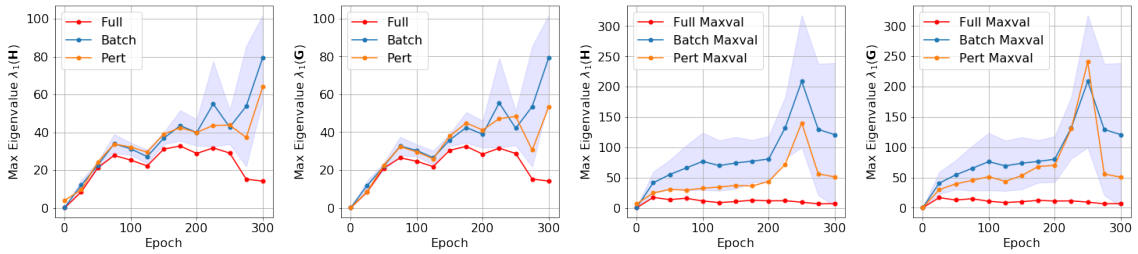


Figure 8: Loss/variance evolution during SGD training for VGG-16 CIFAR-100. Learning Rate schedule specified in Sec 8.



(a) KFAC $\lambda_1(\boldsymbol{H})$ (b) KFAC $\lambda_1(\boldsymbol{G})$ (c) SGD $\lambda_1(\boldsymbol{H})$ (d) SGD $\lambda_1(\boldsymbol{G})$

Figure 9: Evolution of the maximal eigenvalue λ_1 for both the Hessian $\mathbf{H}$ and the GGN matrix $\mathbf{G}$, during SGD and KFAC training on VGG-16 using the CIFAR-100 data set. Full, Batch and Pert refer to the full, batch and the theoretically predicted Hessian eigenvalues respectively. The initial learning rate used is $\alpha = 0.05$, with momentum $\rho = 0.9$ and weight decay 0.0005 for SGD and using $\alpha = 0.003$ and decoupled weight decay 0.01 for KFAC, both using the learning rate schedule in Section 8.

8.2.1 How important is stochasticity?

The batch Hessian extremal eigenvalues have a large variance. This is to be expected, as our results are in the limit of $P, B \to \infty$ and corrections for finite B scale as $B^{-1/4}$ for matrices with finite 4th moments (Bai, 2008), which is $\approx 30\%$ for $B = 128$. Both the theoretical results from the additive noise process (Theorem 4) and multiplicative noise process (Theorem 7) are within 1 standard deviation from the true result. They both follow the increase in variance of the Hessian in Figure 8. We note that the multiplicative noise process provides a better fit. Recent work shows the Hessian outliers to be attributable to the GGN matrix component of the spectrum (Papyan, 2018). Hence a positive semi-definite noise process, tailored to the GGN matrix, would be expected to better estimate the outlier perturbations due to mini-batching - which we observe.

9. Test Accuracy and Movement in the Loss Surface

Given that a major contribution of this paper considers how to scale learning rates as a function of batch size, it is worth taking a moment to consider how learning rates (and their schedules) are often chosen. Typically, whilst many learning rates will lead to good training error, there may be significant differences in validation or test error. Hence, we dedicate a section to better understand choices in learning rates that aid low validation/test errors, i.e. the ability of the solution to generalise.

In this section we bring together intuitions on generalisation, flat minima and distance from the initialisation point. We argue that in the case that there are exponentially many local minima very close in error to that of the global minimum on the training set, similar curves on the validation not training set may give greater ability to discern whether we are appropriately scaling our learning rates with batch size. We argue from our theory that predicts the scaling of Hessian eigenvalues; that if we want to escape similarly sharp minima into similarly flat minima (by using a large learning rate), we need to scale our learning rates by the decrease/increase in sharpness resulting from our increase/decrease in sub-sampling

respectively. How to scale the learning rate with batch size forms the study of our next sections.

Large Learning Rates and their Uses Large learning rates have been shown to induce implicit regularisation, observed in Li et al. (2019). In contrast, too small learning rates have been shown to lead to poor generalisation (Jastrzębski et al., 2017; Berrada et al., 2018). We show an example in Figure 10 where a smaller learning rate for Adam quickly trains worse but generalises better. This is despite the fact that we train with no weight decay, hence there is no confounding $(1 - \alpha\gamma)$ decay factor which depends on the learning rate. Therefore, *learning the largest stable learning rate is an important practical question for neural network training.* There is no definitive answer on why large learning rates seem to correlate with

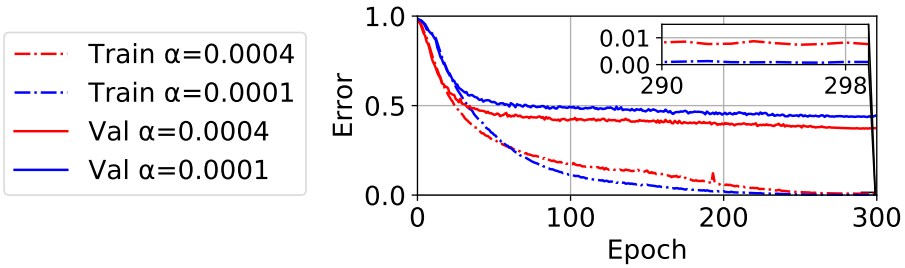
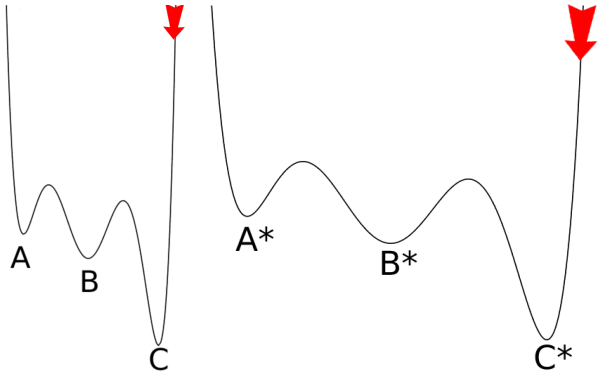


Figure 10: **Smaller learning rates train faster but generalise worse** Training and Validation Error on the VGG-16 on CIFAR-100 with different initial learning rates following the same learning rate schedule with no weight decay.

better generalisation and this topic remains an active area of research. However, one potential intuitive explanation for this phenomenon is that many minima, equivalently or similarly deep in the training loss surface, may have different characteristics on the true loss surface. This is indicated by performance on the validation or test set, which can be considered unbiased estimates of the True Risk/Error. Such minima might be "flatter", generalising better under both Bayesian and minimum description length arguments (Hochreiter and Schmidhuber, 1997; Jastrzębski et al., 2017; Dinh et al., 2017). These arguments can be extended to include parameterisation invariance (Tsuzuku et al., 2020), which makes the correlation between sharpness and test accuracy more robust. In this case, the practice of large learning rate SGD (or any other optimiser) can be viewed as simulated annealing, where we move around the loss surface, limiting our ability to be trapped early on in a sharp local minimum, as the maximal sharpness of the minimum in which we can be trapped is inversely proportional to our learning rate (Wu et al., 2018). Another conjecture, from Hoffer et al. (2017), considers minima which have a greater distance from the initialisation surface, to generalise better. We visualise both of these concepts in Figure 11a, which can be considered a one dimensional slice in the high dimensional surface. If we start with a small learning rate, we settle in minimum C, which is deep (i.e low training loss) but sharp and close to the initialisation point (shown as a red arrow), indicating potentially poor generalisation. If instead we start with a sufficiently large learning rate we can potentially escape such a minimum and with sufficient decay later in training end in minima B or A, which are both flatter and further from the origin. We show evidence that these phenomena are relevant to



(a) Small B (b) Large B

α	Train Acc	Val Acc	Test Acc	$\|\|\Delta_{init}\|\|$
0.08	99.7%	64.35%	65.36%	145.76
0.028	99.89%	61.08%	62.45%	67.44

(c) **Larger distances from initialisation in weight space give solutions of greater generalisation.** Various Learning Rates α for $B = 1024$, with corresponding val/test accuracies and L_2 distance from initialisation for CIFAR-100 on the VGG-16.

Figure 11: Transformed Test Set Surface, going from sharper to flatter minima with an increase in Batch Size. Larger learning rates are more able to escape local poor quality minima which are close to the initialisation.

deep learning in Figure 11*c*, where we show a larger learning rate variant, trains worse, but tests and validates better and has a greater distance from the initialisation point.

9.1 Validation Error Curves as a Proxy for Trajectories in the Loss Surface

When increasing the batch size, our Theorems 4 and 7 indicate that the sharpness of the loss landscape decreases at all points in weightspace. To have a similar trajectory in weight space, where we avoid the "transformed" sharp minimum C^*, shown in Figure 11b, we must move with a larger learning rate since the extremal and bulk edge eigenvalues have decreased in size. Note that depending on whether we mainly move in outlier or bulk directions, the transformations will vary either linearly or with the square root of the batch size respectively.

Since DNNs have been shown to easily fit completely random data (Zhang et al., 2021) and have exponentially local minima in the training loss (Choromanska et al., 2015a), which are close to the global minimum of training loss, there are likely to be many regions of low training error/loss. Hence, when scaling the learning rate in-equivalently with batch size, we could be taking very different paths in weight space, moving through very different minima and still end up with very similar training loss/error curves. Here by in-equivalently we mean as per our example not escaping from minima C/C^* into minimum A/A^* but instead to B/B^*. Hence, we can consider that validation/testing error curves and NOT training error curves should serve as good proxies to identify whether similar trajectories (moving into and through minima of similar sharpness) are being taken along the multi-modal surface.

Note on Trajectory Stochasticity: We note that, since we consider trajectories in expectation, we now discuss whether trajectory stochasticity affects the core arguments presented in this paper. Deep learning initialisation with different random seeds (and hence different starting points and different gradient updates) leads to very similar validation performance. As shown in Appendix C.1. This is not the case, for example, with different learning rates. We thus consider the trajectory in expectation to be the critical factor and not the stochasticity.

Experimental Validation: We show in Figure 11*c* an example of a VGG-16 network on CIFAR-100, trained with different learning rates and a common batch size of $B = 1024$. Despite near identical training performance across the ensemble (noting though that the lower learning rate variant appears to train better), the validation and test accuracies differ significantly with initialisation distance (again noting that these are higher for the higher learning rate variant). We further show in Appendix F that, for this network and a linear scaling relation (which we derive for SGD in the subsequent section) that the distance in L2 norm between initialisation and final solution remains stable across scaling, as does the final testing error and its profile.

Practical Consequences: Whilst we present both training and validation metrics in our experiments, the setup of the experiments, which uses data augmentation, an initially large learning rate (followed by a drop) and often non-zero weight decay, is specifically chosen to provide a low test error. Predicting the learning rates required to achieve similar validation error trajectories, for a given batch size, is key as opposed to finding trajectories leading to similar training error. We show that different learning rates can give rise to (largely) indistinguishable training characteristics unless divergence occurs.

9.2 Experimental Design

Given that neural networks can be trained with a wide variety of schedules, which traverse the loss landscape in very different ways, we need an experimental design which allows us to discern, whether two trajectories in weight space are "similar" and hence whether a proposed scaling rule "works". Having already argued that learning the largest possible initial learning rate is a practical problem for deep learning as such schedules aid deep learning generalisation and that trajectories in the validation error are more meaningful to measure loss trajectory movements, we need to be clear with what we mean by "largest". We find that for the VGG (Simonyan and Zisserman, 2014) networks, without batch normalisation and weight decay, there exists a learning rate value for a given schedule (we use flat and then linear decay) above which (to a certain precision in the grid search) the loss value returns NaN and training breaks. This serves as our definition of maximum and hence for this reason in our experiments we consider the VGG-16 as our reference network. As discussed previously, due to the interaction of batch normalisation with curvature and the lack of explicit treatment in our work on batch normalisation, this network serves as an ideal testing ground, but we conjecture our scaling rules to hold more generally and give preliminary evidence for this. For other networks, including batch normalisation (Ioffe and Szegedy, 2015) and residual layers (He et al., 2016), we find that there exist learning rates above which training and testing both suffer and so we use a working "maximum" which is the largest learning rate which gives a good validation error, close to what is used in practice.

10. SGD learning rates as a function of batch size

One key practical application of Section 3 for neural network training is its implications for learning rates as we alter the batch size. Where weight decay is used, the value of $\gamma = 0.0005$ is employed, giving the best validation performance on the grid of $[0, 0.0001, 0.0005, 0.001]$ and representing a common practical starting choice.

10.1 Finding the Maximal Allowable Learning Rate

The change in batch loss, to second-order approximation, for a small step in the direction of the gradient is given by

$$\delta L_{batch}(\boldsymbol{w} - \alpha \nabla L) = -\alpha ||\boldsymbol{g}(\boldsymbol{w})||^2 \left(1 - \frac{\alpha \sum_i^P \lambda_i ||\boldsymbol{\phi}_i \boldsymbol{g}(\boldsymbol{w})||^2}{2}\right) \leq -\alpha ||\boldsymbol{g}(\boldsymbol{w})||^2 \left(1 - \frac{\alpha \lambda_1}{2}\right). \tag{24}$$

Typically this bound is derived for the deterministic full gradient case (Nesterov, 2013) and hence $\alpha < 2/\lambda_1$. In our case, since the batch is stochastic and hence the gradient and Hessian (and therefore its extremal eigenvalues) are also stochastic, this bound holds in expectation i.e. $\mathbb{E}(\delta L_{batch}(\boldsymbol{w} - \alpha \nabla L))$. It is worth noting that in the framework set out by this paper, the Kolmogorov limit, where $P, B, N \to \infty$ but have constant ratios, that as is typical in the random matrix theory literature, the result concentrates around its mean. Hence the discussion of whether the bound holds in expectation or not is superflous unless we explicitly exit this regime. Here, $\lambda_1(\boldsymbol{H}_{batch})$ is the largest eigenvalue of the batch Hessian (in expectation) which, along with all outlier eigenvalues of the batch Hessian, are given by Theorem 4.

A key term in Equation 24 is the overlap between the eigenvectors and the stochastic gradient, shown to be large in practice (Ghorbani et al., 2019; Gur-Ari et al., 2018). This indicates that the outlier broadening effect predicted by our framework (when there are well separated outliers[7]), i.e $\lambda_i * \approx \lambda_i + P\sigma^2/\mathfrak{b}\lambda_i$, is relevant to determining the maximal allowed learning rate. This follows as the bracketed term in Equation 24 can be written as $(1 - \frac{\alpha |\boldsymbol{g}(\boldsymbol{w})|^2}{2} \sum_i^P \lambda_i \beta_i^2)$. Hence, if $\sum_i^k \beta_i^2 \approx 1$ (where k is the number of outliers) and noting that all outliers scale in a similar way, the result in expectation is similar to that of the expectation of the upper bound. This can be seen in the case where the broadening term dominates the value of the outliers from the empirical Hessian, i.e.

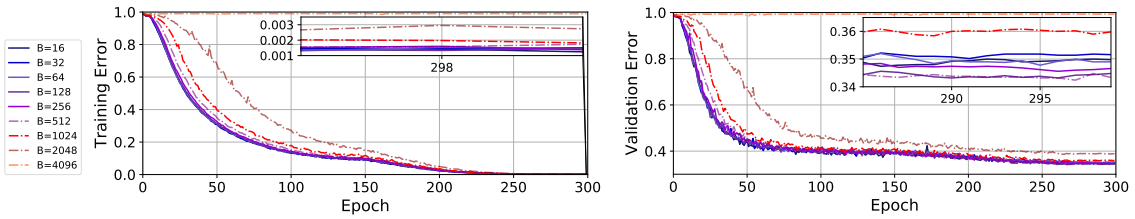
$$1 - \frac{\alpha |\boldsymbol{g}(\boldsymbol{w})|^2}{2} \sum_i \beta_i^2 \left(\lambda_i + \frac{P\sigma^2}{\mathfrak{b}\lambda_i}\right) \approx 1 - \frac{P\sigma^2 \alpha |\boldsymbol{g}(\boldsymbol{w})|^2}{2\mathfrak{b}} \sum_i \frac{\beta_i^2}{\lambda_i}. \tag{25}$$

Note that, if we want to consider the difference between the batch and true Hessian, we would have B instead of $\mathfrak{b}$, where $\mathfrak{b} > B$ and for $B \ll N$ $\mathfrak{b} \approx B$. We observe outliers in all our experiments, as shown in Figures 6 and 7, which is consistent with previous literature (Ghorbani et al., 2019; Papyan, 2018) and motivated in Section 13.

For small batch sizes the maximal learning rate is proportional to the batch size. As the largest allowable learning rate as shown by Equation 24 is $\propto 1/\lambda_1 *$ and $\lambda_1 * = \lambda_1 + P\sigma^2/B\lambda_1$ for very small batch sizes this $\approx P\sigma^2/B\lambda_1$, hence increasing the batch size allows a proportional increase in the maximal learning rate. This holds until the first term λ_1 in Theorem 4 is no longer negligible in comparison to the latter, $P\sigma^2/B\lambda_1$. Thereafter we cross over into the regime where, although the spectral norm still decreases with batch size, it asymptotically reaches its minimal value λ_1. Hence the learning rate cannot be appreciably increased despite using larger batch sizes. To validate this empirically, we train the VGG-16 on CIFAR-100, finding the maximal learning rate at which the network trains for $B = 128$.

7. If there are no outliers, we expect the largest eigenvalue to decrease as the square root of the batch size.

We then increase/decrease the batch size by factors of 2, proportionally scaling the learning rate. We plot the results in Figure 12. The training and validation accuracy remains stable



(a) Training Error against Epoch

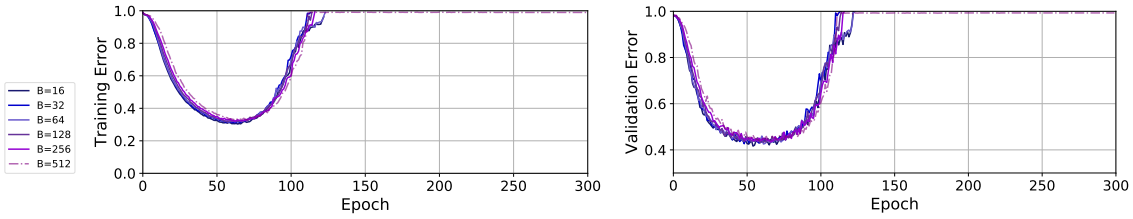
(b) Validation Error against Epoch

Figure 12: **Linear scaling is consistent up to a threshold.** Training and Validation error of the VGG-16 architecture, without batch normalisation (BN) on CIFAR-100, with no weight decay $\gamma = 0$ and initial learning rate $\alpha_0 = \frac{0.01B}{128}$

for all batch size values, until a small drop for $B = 1024$, a larger drop still for $B = 2048$ and for $B = 4096$ we see no training.

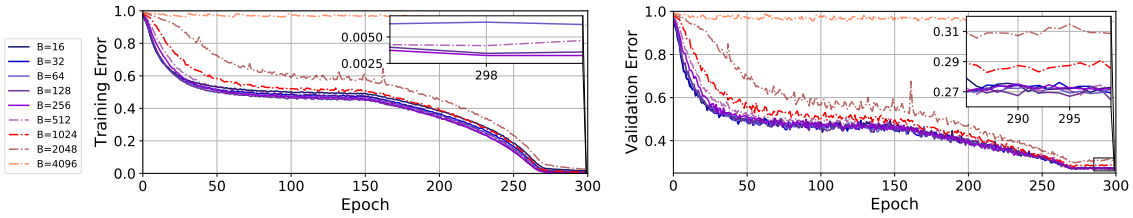
One can get away with large initial learning rates. Another theoretical prediction is that, if the Hessian variance increases during training (as observed in Figure 8), large learning rates which initially rapidly decrease the loss could become unstable later in training. This follows because we want to ensure, at every point in training, that the batch loss does not increase. The largest allowable learning rate which, in expectation, does not increase the batch loss is inversely proportional to the sum of the full Hessian (which is deterministic for a point in weight space) and a term proportional to the variance of the elements of the fluctuations matrix. If the variance of the Hessian (which uniquely determines under the conditions of our model the variance of the fluctuations matrix elements) increases, then we expect the largest allowable learning rate to decrease. Since, in practice, we note that the variance of the Hessian increases during training, we expect for these experiments to be able to start with a larger learning rates. This has also been noted in Lewkowycz et al. (2020). To see this, we run the same experiment but this time use twice the maximal allowed learning rate. We observe in Figure 13a that initially the loss decreases rapidly (far faster than in the smaller learning rate alternative in Figure 12a), but that soon the training becomes unstable and diverges. This indicates that the practice of starting with an initially large learning rate and decaying it is well justified in terms of stability implied by the batch Hessian at least for our experiments.

Our linear scaling rule seems to hold generally for SGD. To highlight the generality of our linear scaling rule, we include batch normalisation (Ioffe and Szegedy, 2015) and weight decay $\gamma = 0.0005$. In this case there is a greater range of permissible learning rates, so we grid search the best learning rate as defined by the validation error for $B = 128$ and use our derived linear scaling rule, as shown in Figure 14, where we observe a similar pattern. We repeat the experiment on the WideResNet-28 $\times$ 10 (Zagoruyko and Komodakis, 2016) on both the CIFAR-100 and ImageNet 32×32 (Chrabaszcz et al., 2017) data sets shown in Figures 15, 16 respectively. Unlike the VGG model without batch normalisation, where unstable trajectories diverge or with batch normalisation do not train. Highly unstable oscillatory



(a) Training Error against Epoch (b) Validation Error against Epoch

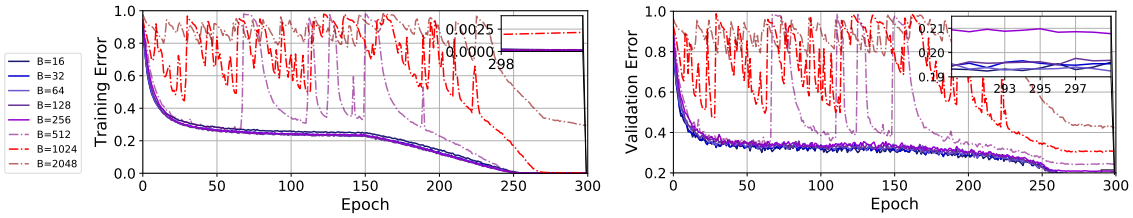
Figure 13: **Consistency holds for a variety of learning rates.** Training and Validation error of the VGG-16 architecture, without batch normalisation (BN) on CIFAR-100, with no weight decay $\gamma = 0$ and initial learning rate $\alpha_0 = \frac{0.02B}{128}$.



(a) Training Error against Epoch (b) Validation Error against Epoch

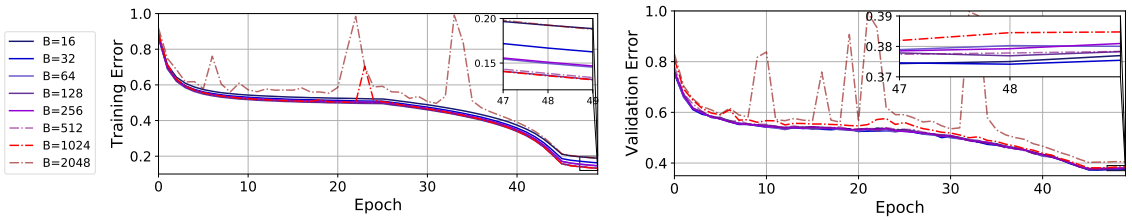
Figure 14: **Linear scaling is consistent up to a threshold.** Training and Validation error of the VGG-16 architecture, with batch normalisation (BN) on CIFAR-100, with no weight decay $\gamma = 5e^{-4}$ and initial learning rate $\alpha_0 = \frac{0.1B}{128}$.

WideResNet trajectories converge with learning rate reduction, however they never reach peak performance. The training and test performance is stable for a variety of learning rates with fixed learning rate to batch size ratio, again strongly supporting the validity of the linear scaling rate rule until a threshold. Note that although not typical practice, we find very similar results for SGD without momentum, as shown in Appendix G.



(a) Training Error against Epoch (b) Validation Error against Epoch

Figure 15: **Consistency holds for a variety of learning rates.** Training and Validation error of the WideResNet-28 $\times$ 10 architecture, with batch normalisation (BN) on CIFAR-100, with weight decay $\gamma = 5e^{-4}$ and initial learning rate $\alpha_0 = \frac{0.1B}{128}$.



(a) Training Error against Epoch

(b) Validation Error against Epoch

Figure 16: **Consistency holds for a variety of learning rates.** Training and Validation error of the WideResNet-28 $\times$ 10 architecture, with batch normalisation (BN) on ImageNet-32, with weight decay $\gamma = 5e^{-4}$ and initial learning rate $\alpha_0 = \frac{0.1B}{128}$.

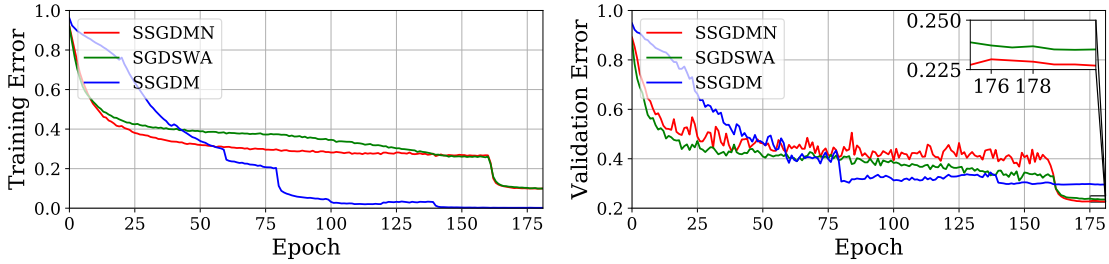
10.2 Estimating the "Optimal" Learning Rate and Momentum from the Spectrum

Our theoretical anaylsis in Section 3 and experiments in Section 8 show that the relevant curvature estimates, when mini-batch training, are not those of the full (or true) Hessian, but rather those of the batch Hessian. This leads to the supposition that we can estimate relevant aspects of the curvature using the Lanczos algorithm in $\mathcal{O}(mPB)$ time during training and, in effect, estimate the optimal learning and momentum rates during training. Note that, since one iteration of SGD is only of cost $\mathcal{O}(PB)$, this procedure needs only to be run irregularly (or alternatively m needs to be kept very small, resulting in poor curvature estimates) for it to be competitive with multiple runs using differing learning rates and/or schedules. As a proof of concept, we run two variants of our approach using the optimality relations for both Polyak and Nesterov learning rates and momenta. Note that these formulas only hold for smooth, strongly convex functions in the deterministic setting. However, we have in effect incorporated the impact of the gradient variance (which increases the value of the Lipshitz constant) into the Lipshitz constant via the increase in expected spectral norm.

$$\alpha_{\text{Polyak}} = \frac{2}{\sqrt{\lambda_1} + \sqrt{\lambda_P}}, \; \alpha_{\text{Nesterov}} = \sqrt{\frac{\lambda_P}{\lambda_1}} \tag{26}$$

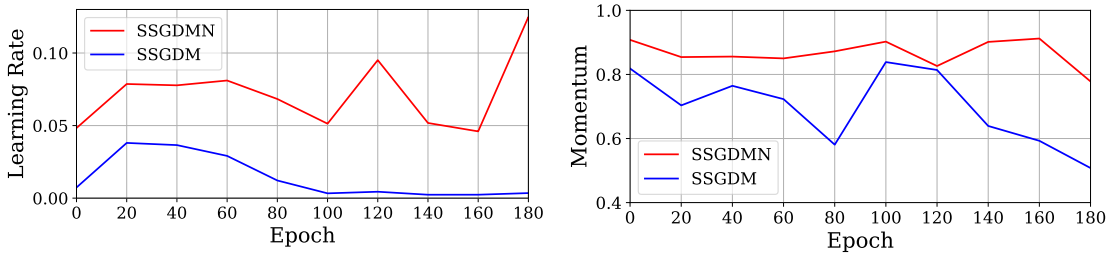
$$\rho_{\text{Polyak}} = \left(\frac{\sqrt{\lambda_1} - \sqrt{\lambda_P}}{\sqrt{\lambda_1} + \sqrt{\lambda_P}} \right)^2, \rho_{\text{Nesterov}} = \left(\frac{\sqrt{\lambda_1} - \sqrt{\lambda_P}}{\sqrt{\lambda_1} + \sqrt{\lambda_P}} \right). \tag{27}$$

Here, the Lipshitz and strong convexity constants are estimated locally using the Lanczos algorithm on the *batch Hessian*. Note that, whilst the Hessian in our experiments has negative spectral mass at all points in weight space (and is hence not strongly convex), we conveniently can use a positive-definite approximation, as is frequently done in the second-order learning literature (Martens and Grosse, 2015; Dauphin et al., 2014). We run a curvature estimate using the Lanczos algorithm, seeded with a random vector every 20 epochs, with iteration number $m = 20$. Neither of these parameters was optimised. Our primary objective is to show that a batch Hessian curvature based approach to learning the learning rate and momentum can be useful out of the box and experimentally reduce and not increase the hyper-parameter burden.



(a) Training Error (b) Validation Error

Figure 17: **Learned Learning Rates seem Competitive with Fine Tuned** PreResNet-110 on the CIFAR-100 data set, with weight decay $\gamma = 5e^{-4}$.



(a) Learning Rate vs Epoch (b) Momentum vs Epoch

Figure 18: Learning Rates and Momenta learned during training for the PreResNet-110 on the CIFAR-100 data set, with weight decay $\gamma = 5e^{-4}$.

Dealing with Stochasticity: Given that, in the stochastic case, all methods must decay the learning rate and/or employ weight averaging we employ the latter (Izmailov et al., 2018) for all methods near the end of training. It is known (Kushner and Yin, 2003) that iterate averaging gives greater robustness to the learning rate schedule and choice, whilst still leading to convergence.

Dealing with Rank Degeneracy: Since the smallest Ritz values are very very close to zero, which would result in a momentum $\rho = 1$, we use a heuristic to remove the smallest Ritz values, whereby if the Ritz value of largest spectral mass has more than 50% of the spectral mass, it is removed and the resulting density renormalised, forming the new spectral density of interest.

We present our results in both training and testing for the PreResNet-110, with weight decay of 0.0005, in Figure 17. Here we compare with the tuned learning-rate schedule used in Izmailov et al. (2018) and described in Section 8. The latter has an initial learning rate set to 0.1. We show the learned learning rates and momenta for both methods in Figure 18. We note that, whilst the Polyak method strongly decays the learning rate, converging fast on the training set, the Nesterov variant, coupled with Nesterov Momentum, keeps the learning rate high, converging only slightly faster than the SGDSWA variant but outperforming in test error at the end. Whilst we don't expect for general non-convex problems, such as deep learning, a method such as this to out-perform all combinations of learning rates and

momentum schedules, it is encouraging that such a cheap estimation approach has significant potential.

11. Square root learning rate scaling for adaptive optimisers with small damping

By considering the change in loss for a generic second-order optimiser, where we precondition the gradients with some approximation of the Hessian $\boldsymbol{B}$, we have

$$L(\boldsymbol{w}_k - \alpha \boldsymbol{B}^{-1}\nabla L(\boldsymbol{w}_k)) - L(\boldsymbol{w}) = \alpha \nabla L(\boldsymbol{w}_k)^T \boldsymbol{B}^{-1}\nabla L(\boldsymbol{w}_k) + \frac{\alpha^2}{2}\nabla L(\boldsymbol{w}_k)^T \boldsymbol{B}^{-1}\boldsymbol{H}\boldsymbol{B}^{-1}\nabla L(\boldsymbol{w}_k).$$

Writing $\boldsymbol{H}_{emp} = \sum_i \lambda_i \boldsymbol{\psi}_i \boldsymbol{\psi}^T$ and writing the noisy estimated eigenvalue/eigenvector pair from the optimiser as $\boldsymbol{B} = \sum_j \eta_j \boldsymbol{\phi}_j \boldsymbol{\phi}_j^T$, making use of orthogonal bases, we have,

$$L(\boldsymbol{w}_{k+1}) - L(\boldsymbol{w}) = \sum_i^P \frac{\alpha_0 |\boldsymbol{\phi}_i^T \nabla L(\boldsymbol{w})|^2}{\eta_i + \delta}\left(1 - \frac{\alpha_0}{2(\eta_i+\delta)}\sum_\mu \lambda_\mu |\boldsymbol{\psi}_u^T \boldsymbol{\phi}_i|^2)\right). \tag{28}$$

This is a more complicated expression than the resulting equation for SGD (Equation 24), as it involves both the eigenvalue/eigenvector pairs of the batch Hessian and that of the preconditioning matrix. Whereas for SGD, movement in the eigenvectors corresponding to the largest eigenvalues result in the greatest increase in loss, *for adaptive optimisers, division by the inverse of the preconditioner eigenvalue means that an increase in the loss function could be due to the optimiser moving direction of lower curvature.*

Potential Boost for Edge Eigenvectors: Consider the simplified case of $|\boldsymbol{\psi}_u^T \boldsymbol{\phi}_i|^2 = \delta_{u,i}$, where we assume perfect eigenvector estimation but potentially imperfect eigenvalue estimation. We then consider an eigenvalue from the batch Hessian which is at the edge of the bulk distribution and thus an outlier. The loss will be larger moving in this "flat" direction iff,

$$\frac{\sqrt{P}\sigma}{\sqrt{\mathfrak{b}}(\eta_i+\delta)} > \frac{\lambda_j + \frac{P\sigma^2}{\mathfrak{b}\lambda_j}}{(\eta_j+\delta)} \quad \text{i.e.} \quad \frac{\eta_j+\delta}{\eta_i+\delta} > \left(\frac{\lambda_j\sqrt{\mathfrak{b}}}{\sqrt{P}\sigma} + \frac{\sqrt{P}\sigma}{\sqrt{\mathfrak{b}}\lambda_j}\right). \tag{29}$$

Hence an under-estimation of the bulk eigenvalue η_i, relative to outlier eigenvalue η_j, combined with a small damping coefficient (typically set at 10^{-8} for Adam) could result in this condition being satisfied. There are many $O(P)$ eigenvalues near the edge of the bulk distribution, compared to the limited number of outliers and hence many edge eigenvalue/eigenvector pairs that need to be well estimated. In general $|\boldsymbol{\psi}_u^T \boldsymbol{\phi}_i|^2 = \alpha_{i,\mu}^2$. Hence in Equation 28 we consider an effective bracketed term of

$$\sum_i^P \frac{\alpha_0 |\boldsymbol{\phi}_i^T \nabla L(\boldsymbol{w})|^2}{\eta_i + \delta}\left(1 - \frac{\alpha_0}{2(\eta_i+\delta)}\sum_\mu \lambda_\mu \alpha_{i,\mu}^2\right) \tag{30}$$

and hence our $\lambda_i = \sum_\mu \lambda_\mu \alpha_{i,\mu}^2$. This reduces to Equation 29 in the case where $\alpha_{i,\mu} = \delta_{i,\mu}$. For random eigenvector estimation $\alpha_{i,\mu}^2 \approx \frac{1}{P}$. For the fully random eigenvector estimation

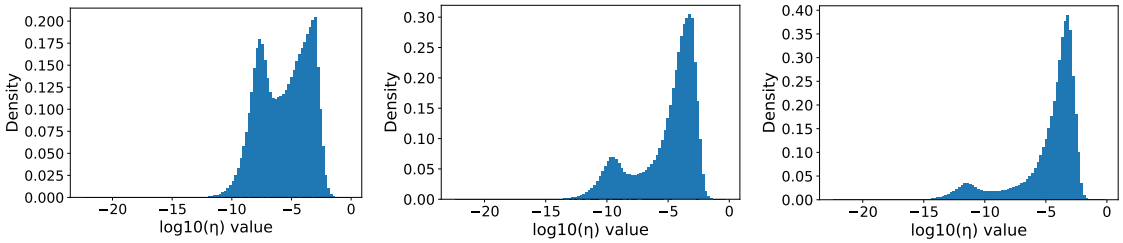
case we move in all directions equally and so our aforementioned phenomenon cannot occur. Hence, the optimiser must be able to discern outlier from bulk spectrum directions. Whilst we do not undertake a full analysis here, it is clear that for k outliers, $\alpha_{i,k} \gg \sqrt{1/P}$. We expect any adaptive gradient optimiser to have some ability to estimate the local curvature. We more fully investigate the performance of Adam to learn the directions of sharpest curvature in Section 11.1.

Necessity of small numerical stability coefficient: In the $\delta \to \infty$ limit, the l.h.s. of Equation 29 is 1, whereas as $\lambda_j > \sqrt{\frac{P}{\mathfrak{b}}\sigma}$ the r.h.s. is > 1. Hence, Equation 29 cannot be satisfied. If we move in all eigendirections equally, then - since an outlier is, by definition, larger in magnitude than eigenvalues at the edge of the bulk - we cannot increase the loss more in a non-outlier direction than in an outlier direction.

Practical Implication: Under the scenario of a small damping, δ, with an adaptive method, we would expect to be able to scale the learning only proportionally to the square root of the batch size, since the bulk eigenvalue distribution scales as the square root of the batch size. Hence,

$$\left(1 - \frac{\alpha_0\sqrt{P}\sigma}{(\eta_i + \delta)\sqrt{\mathfrak{b}}}\right) > 0 \therefore \alpha_0 < \frac{\sqrt{\mathfrak{b}}\kappa}{\sqrt{P}\sigma} \leq \frac{\sqrt{\mathfrak{b}}(\eta_i + \delta)}{\sqrt{P}\sigma}, \tag{31}$$

where $\kappa = \eta_{min} + \delta$ and η_{min} is the worst curvature estimate (transformed into the appropriate basis) of a bulk edge eigenvector. Note since the eigenvectors of the bulk edge all transform as $\propto \sqrt{\mathfrak{b}}$ we can simply absorb the constant into κ. Note further, that for small enough batch size - as the outlier eigenvalues scale proportionally with $\frac{1}{\mathfrak{b}}$ whereas the bulk distribution only grows proportional to $\sqrt{\frac{1}{\mathfrak{b}}}$ - we expect the condition to become harder to fulfil. This means that our misestimation of the bulk eigenvalue/eigenvector pairs needs to increase relative to the outliers in the event of smaller batch sizes. Hence, for very small batch sizes, the scaling could revert to being linear and the square root rule could break down. In order



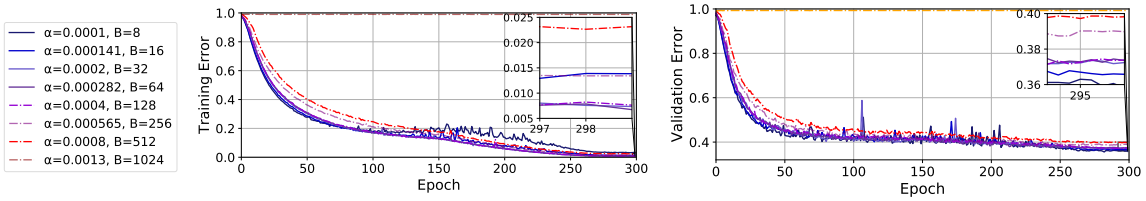
(a) Epoch 25 (b) Epoch 50 (c) Epoch 75

Figure 19: **Huge Variation in Scaling Coefficients for Adam.** Density of pseudo eigenvalues η_i learned during training a VGG-16 on CIFAR-100 using the Adam optimiser for different epoch values, for $\alpha = 0.0004, \gamma = 0$ with a linear decay schedule from Section 8.

to verify that the necessary conditions hold in the commonly used Adam optimiser for such a square root scaling to occur. We investigate the implied curvature eigenvalues η_i from the Adam state dictionary (Chaudhari et al., 2016) in the diagonal basis. We plot the results for

different epochs in Figure 19. Note the huge range in value of η. With a maximum of ≈ 0.6 and a practical minimum of 10^{-8} set by the damping coefficient.

In order to put this derived scaling rule to the test, we run experiments similar to those of Section 10. We find the maximal initial learning rate for the VGG-16 on CIFAR-100 with no weight decay $\gamma = 0$, which stably trains with the Adam optimiser. We use an initial learning rate of $\alpha_0 = 0.0004$ for a batch size of $B = 128$ and then complete a linear learning rate decay schedule, as detailed in Section 8. We then scale the learning rate with the square root of the batch size in either direction and plot the results. We drop the batch-size to 8, to test the limits of our theory. The results are shown in Figure 20. We note excellent agreement down

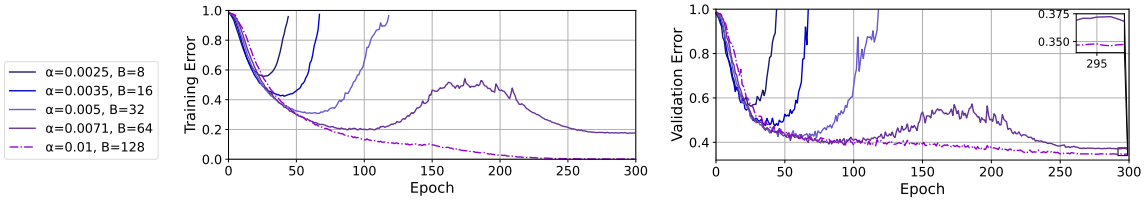


(a) Training Error against Epoch

(b) Validation Error against Epoch

Figure 20: **Square root scaling for adaptive optimisers is consistent up to a threshold.** Training and Validation error of the VGG-16 architecture, without batch normalisation (BN) on CIFAR-100, with no weight decay $\gamma = 0$ and initial learning rate $\alpha_0 = \frac{0.004\sqrt{B}}{\sqrt{128}}$, which varies as a function of batch size B.

to $B = 16$, with very small differences between training curves and validation performance. There is a slight instability in training for $B = 8$, potentially indicating a regime where broadening of the outlier eigenvalues dominates the mis-estimation of the bulk distribution. Note that, when reducing the batch size, reducing the learning rate using square root scaling is a far more aggressive reduction schedule and hence, should the appropriate scaling be linear, training would quickly fail. As is shown for the SGD case, running the same learning rate of 0.01 (from Section 10) and reducing the learning rate using square-root scaling, leads to poor training and divergence, as shown in Figure 21.

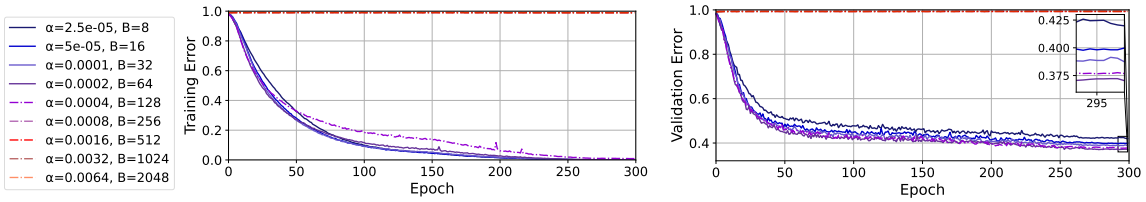


(a) Training Error against Epoch

(b) Validation Error against Epoch

Figure 21: **Square root scaling for SGD does not hold.** Training and Validation error of the VGG-16 architecture, without batch normalisation (BN) on CIFAR-100, with no weight decay $\gamma = 0$ and initial learning rate $\alpha_0 = \frac{0.01\sqrt{B}}{\sqrt{128}}$.

We plot results from attempting to use the linear scaling rule for Adam in Figure 22. Note that there is rapid divergence upon using a doubled batch size. Furthermore, we



(a) Training Error against Epoch

(b) Validation Error against Epoch

Figure 22: **Linear scaling rate does not hold for Adam.** Training and Validation error of the VGG-16 architecture, without batch normalisation (BN) on CIFAR-100, with no weight decay $\gamma = 0$ and initial learning rate $\alpha_0 = \frac{0.0004B}{128}$, which varies as a function of batch size B.

note that experiments with a reduced learning rate and lower batch size show different training loss profiles to that of the the initial setup (shown as the only dotted line that converges). Furthermore, upon inspecting the test error, we note that, unlike the case of square root scaling (which produces consistent test error estimates throughout the range), we see significant ($38.1\% \pm 0.4$, $37.6\% \pm 0.64$ $38\% \pm 0.25$ $40.1\% \pm 0.3$, $42.7\% \pm 0.1$) increases in test error as we reduce the learning rate. We argue that, in expectation, the main effect of sub-sampling is a broadening of the spectrum (increasing eigenvalue magnitude) at all points in weight space. Hence, for an equivalent trajectory in expectation in weight space as we change the batch size, we would expect to scale the learning rate as the inverse of this increase in eigenvalue magnitude. Hence large differences in test error suggest that we are not traversing the surface in an equivalent way and hence not settling to a minimum of similar sharpness/distance from initialisation. Note that it is, in general, always possible to train the network with a lower learning rate. Indeed, in many such cases the training curves are almost indistinguishable. However, we find the test errors are often significantly worse, indicating an in-equivalence in trajectory traversal. We note that this is expected from our previous argument, in Section 9. There we argue that different learning rate schedules induce different trajectories across the loss surface and lead to minima of differing curvature. The latter, despite having similar training error, often have very different validation and test errors. We show this explicitly in Appendix C.

We explicitly compare the linear and square root learning rate as a function of batch size prescriptions side by side in Figure 24. As we increase the batch size, we see that the linear prescription fails to train, whereas the square root prescription trains but to a lower performance. As we decrease the batch size below $B = 64$ we see the validation error constantly increase in the linear prescription, indicating that we are not appropriate exploring the loss surface. However, the square root prescription is stable and consistent until $B = 16$, when it starts to exhibit some instabilities, yet ending on statistically significant ($> 0.5\%$) improved validation error, implying that we are exploring the loss surface more aggressively than the baseline at $B = 128$. This is to be expected, because according to our Theorem 4, in the low batch limit outliers will dominate the bulk edge (by a factor $\frac{1}{\sqrt{B}}$). Hence to keep very similar trajectories we would need to bring the prescription closer to linear as we significantly reduce the batch size. We caveat that whilst this experiment gives good evidence the relevance of the square root rule for adaptive optimisers such as Adam, it is

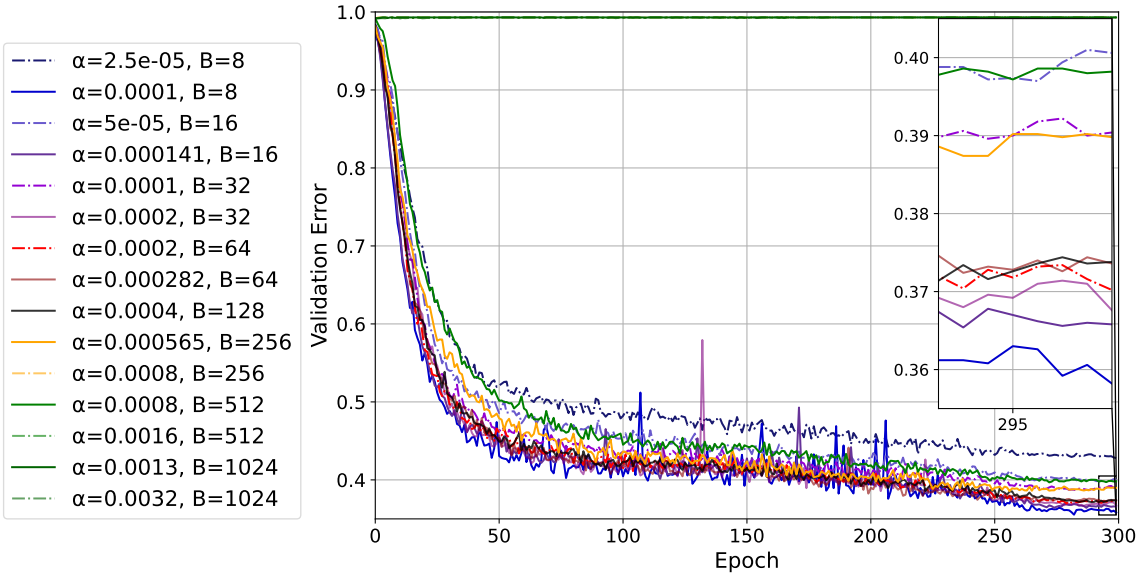


Figure 23: Training Error against Epoch

Figure 24: **Linear learning rate scaling rule is less consistent than the square root rule.** Validation error of the VGG-16 architecture, without batch normalisation (BN) on CIFAR-100, with no weight decay $\gamma = 0$ and initial learning rates $\alpha_0 = \frac{0.0004B}{128}$ and $\alpha_0 = \frac{0.0004\sqrt{B}}{\sqrt{128}}$, which varies as a function of batch size B. Best viewed in colour.

possible that the result will not hold for alternative data sets, or networks architectures and more experiments are needed to establish its superiority. In particular commonly employed practices such as batch normalisation or decoupled weight decay, which will have a non trivial effect on the spectrum could have a significant impact on scaling relations. We include a more complicated experiment, involving residual connections, decoupled weight decay and batch normalisation on the downsampled ImageNet data set in Appendix H which indicates that even the linear scaling rate is too aggressive to achieve consistent training and validation curves. We leave a fuller analysis of this significant deviation to future work.

11.1 Validating the Accurate Estimation of the Large Eigenvalue/Eigenvector Pairs Assumption

The past section relied heavily on the notion that, for adaptive optimisers, the sharpest eigenvalue/eigenvector pairs of the batch Hessian were better estimated than those at the edge of the bulk. Intuitively, it seems reasonable that these "pure noise" eigenvectors (which are distributed on the unit sphere) might change rapidly from iteration to iteration and since they occupy a sizeable fraction of the loss landscape (compared to the small number of outliers), some or many of them might be severely underestimated. However, given that it is unclear to what extent the empirical Fisher approximation (Kunstner et al., 2019) faithfully approximates local curvature and noting that Adam serves as a running diagonal approximation to the empirical Fisher, it is unclear whether Adam learns any information

about the top eigenvalue/eigenvector pairs of the batch Hessian. In order to test this empirically, we run Adam on the VGG-16 using CIFAR-100 with no weight decay and set the damping coefficient $\delta = 1$. We use the linear learning rate schedule from Section 8. Using the quadratic approximation from Equation 28 we expect the largest possible learning rate before batch loss increases to depend upon the ability, via $\boldsymbol{B}$, to estimate the largest eigenvalue/eigenvector pairs of $\boldsymbol{H}$[8]. Hence, the largest learning rate achievable is a direct measure of the estimation accuracy of the sharpest eigenvalue/eigenvector pairs of the batch Hessian. We search for the highest stable learning rate, α, along a logarithmic grid, with end points $\in (0.01, 1)$. All methods incorporate a momentum of $\rho = 0.9$. We find that the largest *stable* rates for SGD, Adam and KFAC are $0.01, 0.12, 0.32$ respectively, indicating that both KFAC and Adam are significantly better able to estimate sharp curvature directions than curvature agnostic SGD. Given that KFAC is a well-known second-order method that uses the Fisher approximation rather than an empirical measure, we believe this experiment indicatives that Adam reliably learns information about the largest eigenvalue/eigenvector pairs. We show the training and testing error curves from these experiments in Figures 25,

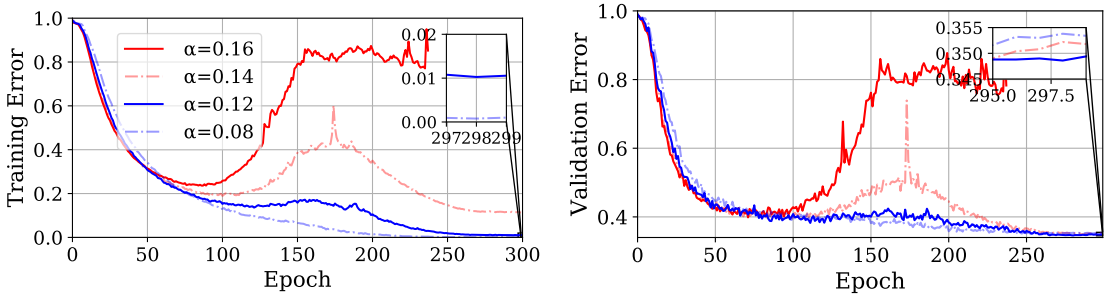


Figure 25: Training/Validation error of Adam using $\delta = 1$ and α on the VGG-16 CIFAR-100

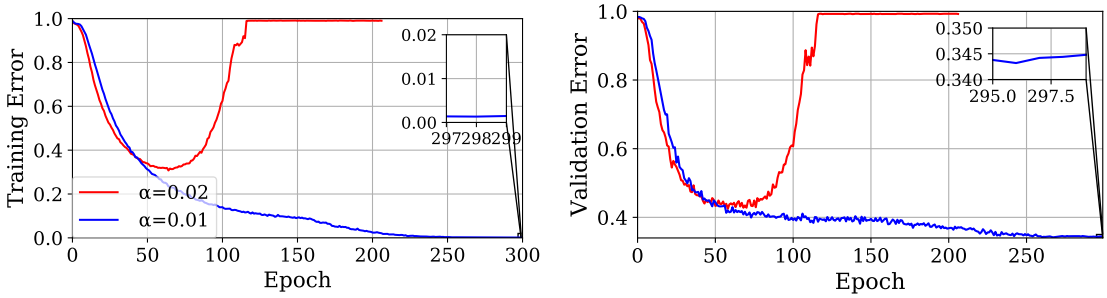


Figure 26: Training/Validation error of SGD using $\delta = 1$ and α on the VGG-16 CIFAR-100

26 and 27 for Adam, SGD and KFAC respectively. For SGD and Adam, overly large learning rates lead to returning to either near random or very low performance. We note that, certain curves for Adam and for all the KFAC curves, have more nuanced form. For KFAC we can even use a learning rate of 1 without running into NaN errors. However, even when annealing this learning rate by a factor of 100 at the end of training, we do not converge in

8. This follows as $\frac{\lambda_i}{\eta_i + 1}$ is largest for large λ_i / η_i

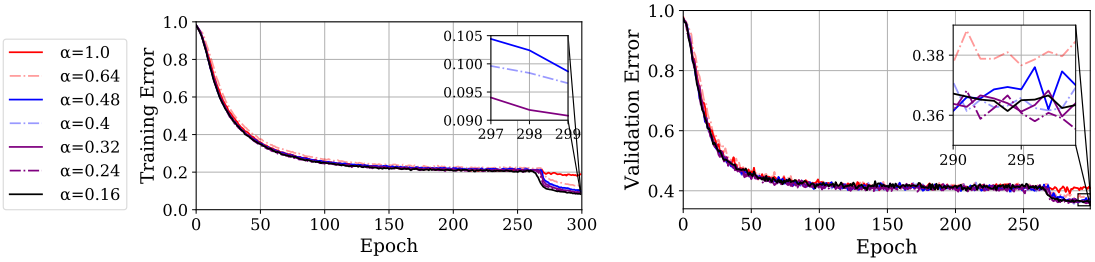


Figure 27: Training/Validation error of KFAC using $\delta = 1$ and α on the VGG-16 CIFAR-100

training or test error. We hence loosely define "stable" as the largest permissible value which allows for training and in the event that a wide range of learning rates are permitted we take the value which gives the best test error.

11.2 Comparison to Gradient Noise

We consider in this section deriving similar scaling relations without a random matrix theory approach. This line of thought is similar to the work in Smith and Le (2017); Smith et al. (2017). Consider a linear approximation to the change in loss from one step of SGD,

$$L(\boldsymbol{w} - \alpha \nabla L(\boldsymbol{w})) - L(\boldsymbol{w}) = -\alpha ||\nabla L(\boldsymbol{w})||^2. \tag{32}$$

For a mini-batch of size B (assuming independent samples), whilst the value of the change in loss remains unchanged in expectation, the variance changes as

$$\mathrm{Var}\left(L(\boldsymbol{w}) - \alpha \left(\frac{1}{B}\sum_i^B \nabla L(\boldsymbol{w})_i\right)\left(\frac{1}{B}\sum_j^B \nabla L(\boldsymbol{w})_j\right)\right)^T \propto \frac{\alpha^2}{B^2}, \tag{33}$$

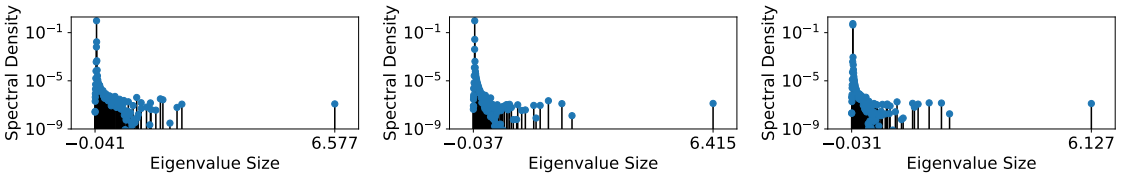
Under such a framework, where we take a linear approximation, we would scale the batch size linearly with the learning rate (within the limits of the approximation) in order to keep the loss variance similar. However, unlike our approach which considers optimal learning rates based on the curvature (and the changes in curvature with batch size), it is somewhat unclear why we would want to keep the variance of the loss equal. Furthermore, unlike our method, for which we can estimate the curvature of the batch from the batch and use this to inform our choice of learning rates and momenta for SGD, the variance of the loss gives no prescription as to which learning rates or momenta we should choose. Furthermore for adaptive methods, the corresponding variance is now

$$\alpha^2 \mathrm{Var}\left(\boldsymbol{B}^{-1}\left(\frac{1}{B}\sum_i^B \nabla L(\boldsymbol{w})_i\right)\left(\frac{1}{B}\sum_j^B \nabla L(\boldsymbol{w})_j\right)^T\right) \propto \tag{34}$$

which is a function of the preconditioning matrix and the covariance of the gradients and from which no clear scaling relationships follow (hence we do not give a scaling in the resulting equation). Considering a spiked random matrix model of the covariance of the gradients, would give similar results to our analysis, but again would require the use of random matrix theory. Whilst it is possible to ignore the effect of the pre-conditioning matrix as in Smith et al. (2017) and argue for a linear prescription, we show in this paper the sub-optimality of this naive approach.

12. True Hessian

We have, to this point, considered the empirical Hessian and the batch Hessian. We here consider the Hessian under the data generating distribution, originally (partially) investigated in Granziol et al. (2018). For finite P and $N \to \infty$, i.e. $q = P/N \to 0$, $|\boldsymbol{\epsilon}(\boldsymbol{w})| \to 0$ the empirical Hessian would become the true Hessian. P, N refer to the parameter count and data set size respectively. Similarly, in this limit, the empirical risk converges almost surely to the true risk, i.e. we eliminate the *generalisation gap*. However, in much deep learning, the network size eclipses the data set size by orders of magnitude.[9] This is similar to considering perturbations between the true covariance matrix and the noisy sample covariance matrix, extensively studied in mathematics and physics (Baik and Silverstein, 2006; Bloemendal et al., 2016b,a). As the true Hessian and true risk are in practice unobservable, we consider whether



(a) Emp-Hessian $N = 50,000$ (b) Aug-Hessian $N = 500,000$ (c) Aug-Hessian $N = 5,000,000$

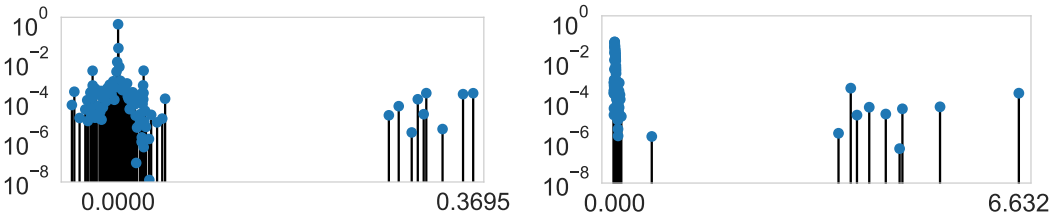
Figure 28: Hessian Spectral Density at epoch 300 , on a VGG-16 on the CIFAR-100 data set, for different amounts of data-augmentation.

the empirical Hessian provides a valid approximation to the true Hessian. We evaluate the Hessian by artificially increasing the number of samples through the use of data-augmentation. We simulate the effect of increasing the data-set size, with random horizontal flips, 4×4 zero padding and random 32×32 crops. We then use the Pearlmutter trick (Pearlmutter, 1994) on the augmented data set, combined with the Lanczos algorithm, to estimate the spectral density. While there is clear dependence between the augmented samples and original samples, intuitively we can consider the augmented data set to be equivalent to an independent set, of larger size than the original data set. This intuition is grounded in the observation that training without augmentation leads to significant performance decreases, similar to reducing the data set size. Specifically, for the VGG-16 without augmentation, we achieve a testing accuracy of 48.8% compared to 72.1% on the CIFAR-100 data set running the same schedule. As shown in Figure 28, the extremal eigenvalues are reduced in size as the sample number is increased, in accordance with Theorem 4. We note from Figure 28 that, despite a factor of 100 in augmentation, the differences between the most augmented and empirical Hessian are slight. There is a slight reduction in the extremal eigenvalues, but otherwise the general shape of the eigenspectrum remains unaffected - implying that the true Hessian may be similar (in its spectral properties) to the empirical Hessian. This is a different conclusion to that reached in (Granziol et al., 2018), where the authors conclude that the true Hessian is very different to that of the empirical Hessian. Note that, were we to consider our batch Hessian to be i.i.d. draws from the data generating distribution, we need only replace $\mathfrak{b}$ in Equation 4 with B, where $\mathfrak{b} > B$ and for $B \ll N$ then $\mathfrak{b} \approx B$.

9. CIFAR data sets, which have $50,000$ examples, are routinely used to train networks with about 50 million parameters.

13. Why do DNN Spectra always have Outliers?

As is evident from Theorems 4 and 7, the scaling of the spectral norm as a function of batch size depends on whether the spectrum has outliers or not. Papyan (2020) uses a formal procedure to attribute various parts of the spectrum to within-class and cross-class covariances. These are then validated experimentally on the VGG-11 architecture with a subsampled CIFAR-10 data set. The work analytically shows, for softmax regression and a k-class Gaussian Mixture Model, that the distance between the outliers (along with the distance between the outliers and the bulk) increases as a function of class separation. Future work can look to extend these results to a 1-hidden layer MLP. Note that, in Figure 29, both at initialisation and training end (where we achieve good class separation and training accuracy), distinct outliers in the spectrum are observed. We also note, at initialisation of



(a) Spectrum at Initialisation

(b) Spectrum at Training End

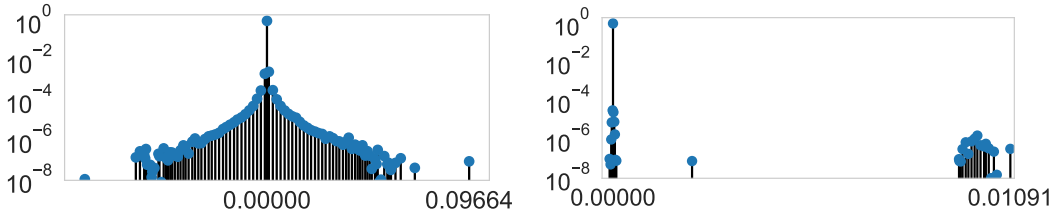
Figure 29: Outlier persistence: Spectrum of a 1-Layer MLP on a 10 class Gaussian Mixture model at Initialisation (where performance is random, at 10%) and End of Training for 100 Epochs with $\alpha = 0.01$ and linear learning rate decay (where the performance is over 95%).

the VGG network in Figure 30a, that we observe some outliers. Following Papyan (2020) we train with an extreme learning rate (we use $\alpha = 0.2$ with $\gamma = 0$ weight decay - compared to best performance values for this network of $\alpha = 0.01$ and no weight decay). We find that this extreme training regime takes the network to a point of no return (i.e. we cannot improve training performance from this point, even by reducing the learning rate and increasing the weight decay factor). We see that, at the end of training, the largest eigenvalues are clustered together to form an outlying continuous spectral density, as shown in Figure 30b. This implies (Papyan, 2020) that there is no class separation. We plan to investigate the apparent need, during training, to retain singleton spectral outliers in future work.

These preliminary experiments, along with the work of Papyan (2020), indicate that for the neural networks in the regime in which we are interested in (i.e. initialisation and training regimens where the network training and testing accuracy continually increase) outliers should be present in the spectrum.

13.1 Regression Spectra

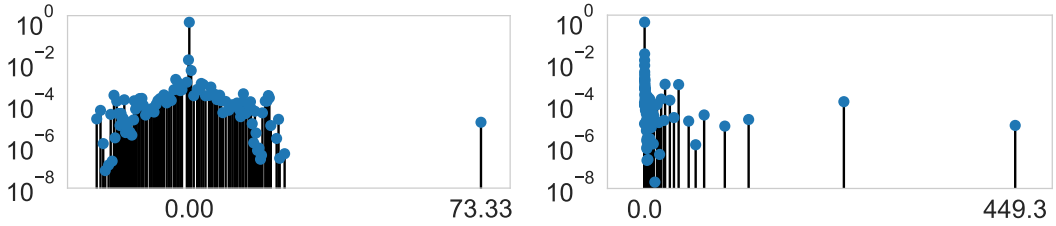
It is interesting to consider whether outliers in the Hessian are not only pervasive across a variety of network architectures, as shown previously, but also are observed across different losses and tasks. Whilst a full study of the spectrum of deep learning problems is beyond the scope of our paper, we consider not just the cross-entropy classification loss function, but a regression problem with L_2 loss. We investigate the Bike data set (Fanaee-T and Gama,



(a) Spectrum at Initialisation (b) Spectrum at Training End

Figure 30: **VGG Network Always contains outlayers**. Spectrum of a 16-Layer VGG Network on the 10 class CIFAR-100 data set, at Initalisation (where performance is random at 1%) and end of Training for 100 Epochs with $\alpha = 0.2, \gamma = 0$ and linear learning rate decay (where the performance remains random and is irrecoverable even with a learning rate drop).

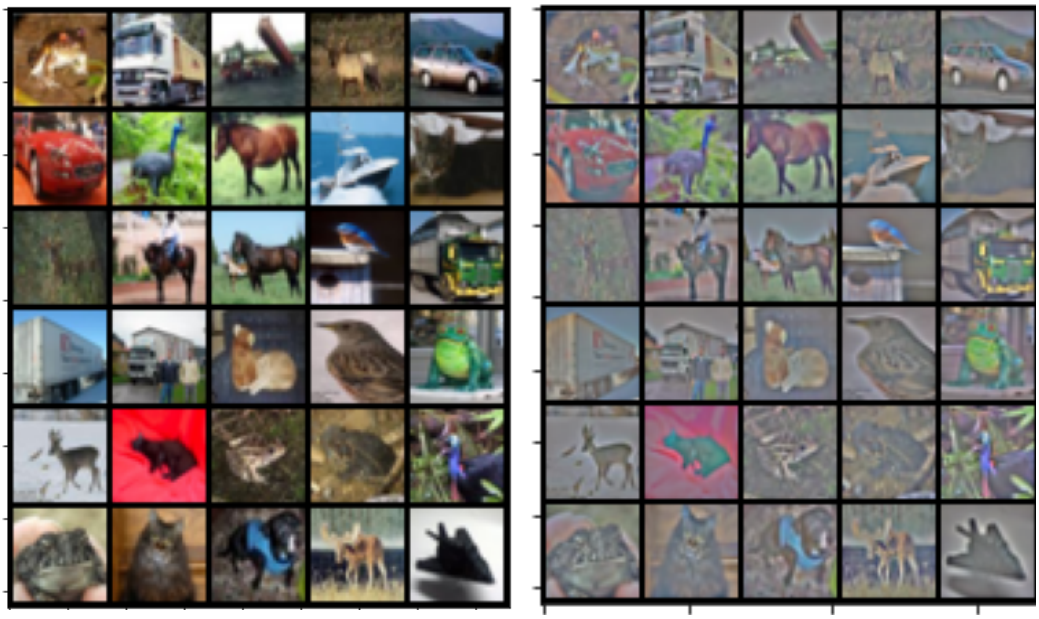
2013), consisting of 13-dimensional feature vectors and a single-dimensional regression target. The architecture of the model has the typical "hour-glass" shape (from 13 inputs to 100 neurons), gradually tapering to a single output. The final test loss (namely the mean squared error) of the trained model is 0.044, which is competitive with baseline results. For example, Wang et al. (2019) report a 0.048 mean squared error using a Gaussian Process model. As



(a) Spectrum at Initialisation (b) Spectrum at End

Figure 31: Spectrum of a 3-Layer MLP Network on the Bike data set, which is a regression (square loss) problem. Initial/final training/testing loss respectively are (0.71/0.66) and (0.037/0.04)

shown in Figure 31, we obtain outliers both at the start of training and at the end. This is intuitively expected. For a single layer multi-layer perceptron, we return to the linear regression problem, for which the Hessian is simply the data-covariance matrix. The outliers of the data covariance matrix are a well studied object in random matrix theory (Bun et al., 2016, 2017; Bai et al., 1996; Bai and Golub, 1997) for which we expect strong correlations and hence outliers. Note that not centering the data covariance matrix is likely to generate at least a single outlier. It is, of course, not clear whether an MLP with several layers, which learns a non-linear combination of features, should preserve this outlier and bulk structure, but at least empirically, in our limited set of results, this seems to be the case.



(a) CIFAR-10 Original

(b) ZCA $\epsilon = 0.1$

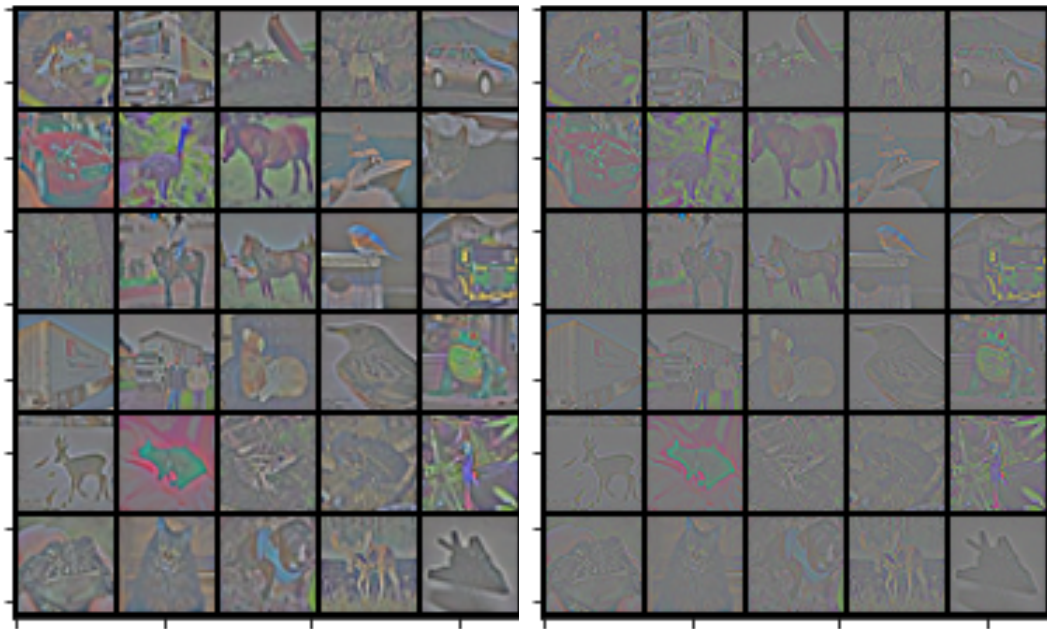
Figure 32: Contrast between original CIFAR-10 images and ZCA whitened versions for a given tolerance value ϵ.

13.2 Can we construct a phase transition in DNN spectra?

Given that our theory (Theorems 4, 7) predicts different regimes and scalings when the largest eigenvalues are within or outside the bulk eigen-distribution, it is of interest to consider if it is possible to construct scenarios in which the outliers can be manipulated (through the use of data pre-processing) to fall into the bulk spectrum. One such method for linear regression is ZCA whitening Bell and Sejnowski (1996). ZCA whitening can be considered as PCA whitening under the constraint that the new data lies as close as possible, in Euclidean distance, to the original data. Since PCA (and ZCA) whitening divides the data matrix by the square root of the eigenvalues of the data covariance matrix, the new data covariance can be seen as "white" i.e. all the eigenvalues have the same magnitude[10]. However, in our problems of interest not only do we not use a square loss function, but we have very deep non-linear neural networks. Hence it is not clear to what extent a priori a whitening of the data covariance matrix will whiten the Hessian of the loss. We note that the eigenvalues of the data covariance matrix may be zero or very small. In the former case, scaling of the covariance may be undefined and in the latter case is likely to cause numerical problems. As is typical for such solvers, we add a small $\epsilon > 0$ to the eigenvalues. The value of ϵ thus determines the strength of the whitening, a lower value inducing a more strongly whitened signal. We show the difference between non-whitened and whitened images in Figure 32. Visually, the images appear as if they have been inverted/greyscaled with a reduced contrast. The severity of the effect is increased as we decrease ϵ as shown in Figure 33, where we

10. White noise has a constant power spectrum, hence the term “whitening”.

successively decrease ϵ from 0.1 by factors of 10. In order to test whether we observe a phase transition over decreasing values of ϵ, we run SGD on CIFAR-10 for 10 epochs, using the linear schedule from Section 8, an initial learning rate of 0.001 and a weight decay of 0.0001. The reason for using such a reduced learning rate is that we find our typical learning rate of 0.05 with weight decay 0.0005 does not train at all. We use such a small number of epochs (10) due to the excessive training times. Indeed, pre-processing of the data covariance matrix is so costly that a single epoch of training is considerably more expensive than the entire 300 epoch training cycle without using ZCA. We note that the purpose of this experiment is not to evaluate performance using ZCA in practice, but simply to consider whether phase transitions occur in which the outliers are subsumed into the bulk spectrum. As shown in

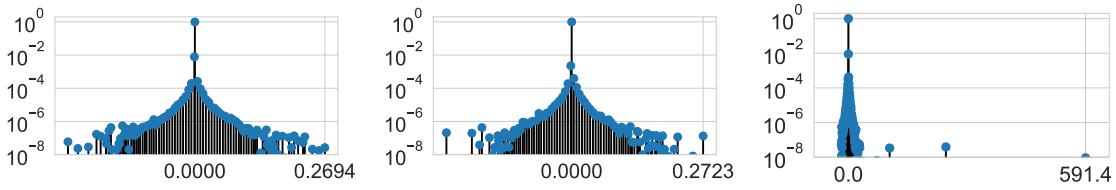


(a) ZCA $\epsilon = 0.01$ (b) ZCA $\epsilon = 0.0001$

Figure 33: ZCA whitened versions for a given tolerance value ϵ

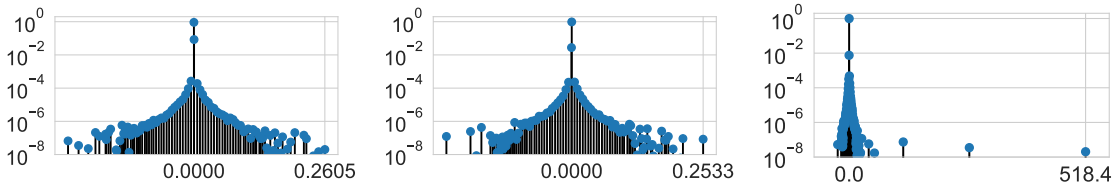
Figures 34 and 35, whilst at the start of training the spectrum is outlier free, that even with ZCA whitening outliers emerge later in training. We start to see evidence of this in Figures 34b and 35b for epoch 1 and this is very clear by epoch 10 (Figures 34c and 35c). Whilst a smaller coefficient of ZCA whitening does seem to tighten the spectrum somewhat at all epochs, this effect is not significant compared to the size of the outlier. We thus conclude experimentally that for Image problems on deep neural networks it does not seem possible to bring outliers back into the bulk through ZCA pre-processing beyond the point of initialisation.

We further investigate the spectrum of a $B = 128$ sub-sample of the data set. Similar to Section 8, we expect the sub-sampled Hessian to be a broadened version of the full data set version. Note that the batch Hessian is what we expect the optimiser to be affected by in mini-batch training and hence the interesting dynamics during training are determined by this matrix and not the full data set Hessian. Interestingly, despite observing some reduction



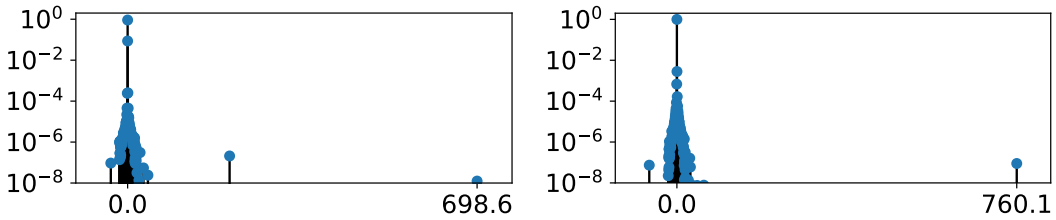
(a) Epoch 0 (b) Epoch 1 (c) Epoch 10

Figure 34: **ZCA whitening removes outliers early in training but they begin to emerge later in training**. Spectrum at various Epochs of the full data set sub-sampled CIFAR-10 data set which is pre-whitened, both in training and during curvature calculation, using ZCA with strength $\epsilon = 0.1$.



(a) Epoch 0 (b) Epoch 1 (c) Epoch 10

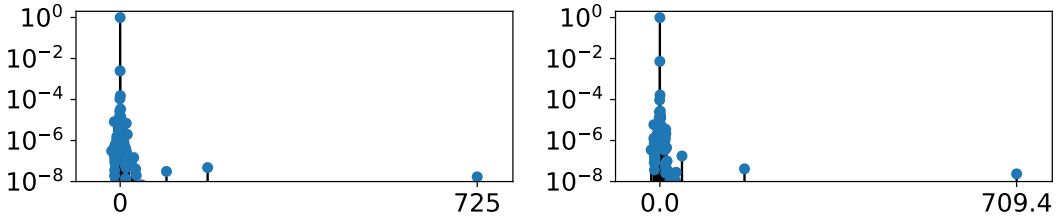
Figure 35: **Outliers remain later in training even with extensive ZCA pre-processing:** Spectrum at various Epochs of the full data set sub-sampled CIFAR-10 data set which is pre-whitened, both in training and during curvature calculation, using ZCA with strength $\epsilon = 0.001$.



(a) Spectrum $\epsilon = 0.1$ (b) Spectrum $\epsilon = 0.01$

Figure 36: **Optimiser sees a very sharp surface during training with ZCA**.Spectrum at Epoch 8 of sub-sampled $B = 128$ CIFAR-10 data set which is pre-whitened, both in training and during curvature calculation, using ZCA with various strengths ϵ.

in the size of the largest outlier by decreasing ϵ - as Figure 36 - we note that later in training (Epoch 8, where we have around 30% accuracy) we see the emergence of very large outliers, well separated from the bulk spectrum. Note that due to the stochasticity of the results we would need to run several samples to get a good mean estimate for the batch spectrum as in the previous section 8. Due to the computational complexity of running these experiments, we omit such analysis. Furthermore, since we find that outliers emerge as soon as we get



(a) Spectrum $\epsilon = 0.001$ (b) Spectrum $\epsilon = 0.0001$

Figure 37: Spectrum at Epoch 8 of sub-sampled $B = 128$ CIFAR-10 data set which is pre-whitened, both in training and during curvature calculation, using ZCA with various strengths ϵ.

class seperation we do not investigate the linear and square root scaling rules using ZCA whitening on image classification experiments.

14. Conclusion

This paper shows that, under a spiked, field-dependent random matrix model, the extremal eigenvalues of the batch Hessian are larger than those of the empirical Hessian. The magnitude of the perturbation is inversely proportional to the batch size if there are well separated outliers in the Hessian spectrum and inversely proportional to the square root if not. The main implications of this work are that up to a threshold: 1) SGD learning rates should be scaled linearly with batch size; 2) Adam learning rates should be scaled with the square root of the batch size; 3)When trying to learn the learning rates and momenta directly from curvature, we should use the batch not empirical Hessian. We extensively validate our predictions and associated implications on the VGG-16 network and CIFAR-100 data set, across various hyper-parameter settings, including weight-decay and batch-normalisation. For SGD, we further validate our prediction on the WideResNet-28 $\times$ 10 on both the CIFAR-100 and ImageNet-32 data sets. Given that our analysis is neither data set nor architecture specific, we expect our results to hold generally outside of our experimental setup. This work can be used to better inform practitioners of how to adapt learning rate schedules for both small and large devices in a principled manner. We further investigate to what extent it is possible to remove outliers from the spectrum of deep neural networks using ZCA whitening. We find that it is possible to remove such outliers at network initialisation, but that they return during training. We find that outliers are generally very pervasive in deep network spectra, including a regression example and even the MLP on Gaussian Mixture Data. We find the latter result to be very intersting in so far that it visually seems strikingly similar to the spectra of deep neural networks on image classification data. This could be interesting future theoretical work.

15. Acknowledgements

The lead author would like to thank Nicholas P Baskerville and Jon Keating for discussions surrounding the context behind this work and Random Matrix Theory, along with Timur

Garipov and Dmitry Vetrov for the opportunity to develop software relevant to this line of research in Moscow. The lead author would further like to thank the Oxford-Man Institute for funding the initial stages of this extensive research direction, Andrew Gittings from the JADE team for extensive computational resources and Hafiz Tiomoko & Xingchen Wan for further discussions and interesting ideas on potential experimental validation. Further thanks must be given to Li Zhenguo for his support and encouragement in pursuing and polishing this research direction, along with Pavel Izmailov who repeatedly emphasised the value behind this work and the need to bring it to publication.

References

Radoslaw Adamczak. On the marchenko-pastur and circular laws for some classes of random matrices with dependent entries. *Electronic Journal of Probability*, 16:1065–1095, 2011.

Zhaojun Bai and Gene H. Golub. Bounds for the Trace of the Inverse and the Determinant of Symmetric Positive Definite Matrices. *Annals of Numerical Mathematics*, 4:29–38, 1997.

Zhaojun Bai, Gark Fahey, and Gene Golub. Some large-scale matrix computation problems. *Journal of Computational and Applied Mathematics*, 74(1-2):71–89, 1996.

Zhi Dong Bai. Convergence rate of expected spectral distributions of large random matrices part i: Wigner matrices. In *Advances In Statistics*, pages 60–83. World Scientific, 2008.

Jinho Baik and Jack W Silverstein. Eigenvalues of large sample covariance matrices of spiked population models. *Journal of multivariate analysis*, 97(6):1382–1408, 2006.

Anthony Bell and Terrence J Sejnowski. Edges are the'independent components' of natural scenes. *Advances in neural information processing systems*, 9, 1996.

Florent Benaych-Georges and Raj Rao Nadakuditi. The eigenvalues and eigenvectors of finite, low rank perturbations of large random matrices. *Advances in Mathematics*, 227(1): 494–521, 2011.

Leonard Berrada, Andrew Zisserman, and M Pawan Kumar. Deep Frank-Wolfe for neural network optimization. *arXiv preprint arXiv:1811.07591*, 2018.

Alex Bloemendal, Antti Knowles, Horng-Tzer Yau, and Jun Yin. On the principal components of sample covariance matrices. *Probability theory and related fields*, 164(1-2):459–552, 2016a.

Alex Bloemendal, Bálint Virág, et al. Limits of spiked random matrices ii. *The Annals of Probability*, 44(4):2726–2769, 2016b.

Raghu Bollapragada, Richard H Byrd, and Jorge Nocedal. Exact and inexact subsampled newton methods for optimization. *IMA Journal of Numerical Analysis*, 39(2):545–578, 2019.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

Stephen P. Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2009.

Joël Bun, Romain Allez, Jean-Philippe Bouchaud, Marc Potters, et al. Rotational invariant estimator for general noisy matrices. *IEEE Trans. Information Theory*, 62(12):7475–7490, 2016.

Joël Bun, Jean-Philippe Bouchaud, and Marc Potters. Cleaning large correlation matrices: tools from random matrix theory. *Physics Reports*, 666:1–109, 2017.

Tony Cai, Jianqing Fan, and Tiefeng Jiang. Distributions of angles in random packing on spheres. *The Journal of Machine Learning Research*, 14(1):1837–1864, 2013.

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. *arXiv preprint arXiv:1611.01838*, 2016.

Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204, 2015a.

Anna Choromanska, Yann LeCun, and Gérard Ben Arous. Open problem: The landscape of the loss surfaces of multilayer networks. In *Conference on Learning Theory*, pages 1756–1760, 2015b.

Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pages 2933–2941, 2014.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1019–1028. JMLR. org, 2017.

Hadi Fanaee-T and Joao Gama. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, pages 1–15, 2013. ISSN 2192-6352. doi: 10.1007/s13748-013-0040-3. URL `[WebLink]`.

Adina Roxana Feier. *Methods of proof in random matrix theory*. PhD thesis, Harvard University, 2012.

Jack Fitzsimons, Diego Granziol, Kurt Cutajar, Michael Osborne, Maurizio Filippone, and Stephen Roberts. Entropic trace estimates for log determinants. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 323–338. Springer, 2017.

Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. Gpytorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems*, pages 7576–7586, 2018.

Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via Hessian eigenvalue density. *arXiv preprint arXiv:1901.10159*, 2019.

Noah Golmant, Nikita Vemuri, Zhewei Yao, Vladimir Feinberg, Amir Gholami, Kai Rothauge, Michael W Mahoney, and Joseph Gonzalez. On the computational inefficiency of large batch sizes for stochastic gradient descent. *arXiv preprint arXiv:1811.12941*, 2018.

Gene H Golub and Gérard Meurant. Matrices, moments and quadrature. *Pitman Research Notes in Mathematics Series*, pages 105–105, 1994.

Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU press, 2012.

F Gotze, AA Naumov, and AN Tikhomirov. Limit theorems for two classes of random matrices with dependent entries. *Theory of Probability & Its Applications*, 59(1):23–39, 2015.

Friedrich Götze, A Naumov, and A Tikhomirov. Semicircle law for a class of random matrices with dependent entries. *arXiv preprint arXiv:1211.0389*, 2012.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Diego Granziol and Stephen Roberts. An information and field theoretic approach to the grand canonical ensemble, 2017.

Diego Granziol, Timur Garipov, Stefan Zohren, Dmitry Vetrov, Stephen Roberts, and Andrew Gordon Wilson. The deep learning limit: are negative neural network eigenvalues just noise? *ICML: Physics in Deep Learning Workshop*, 2018.

Diego Granziol, Xingchen Wan, and Timur. Garipov. MLRG deep curvature. *arXiv preprint arXiv:1912.09656*, 2019.

Diego Granziol, Xingchen Wan, Samuel Albanie, and Stephen Roberts. Iterative averaging in the quest for best test error. *arXiv preprint arXiv:2003.01247*, 2020.

Guy Gur-Ari, Daniel A Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

Walid Hachem, Philippe Loubaton, Jamal Najim, and Pascal Vallet. On bilinear forms based on the resolvent of large random matrices. In *Annales de l'IHP Probabilités et statistiques*, volume 49, pages 36–63, 2013.

Nicholas JA Harvey, Christopher Liaw, Yaniv Plan, and Sikander Randhawa. Tight analyses for non-smooth stochastic gradient descent. In *Conference on Learning Theory*, pages 1579–1613. PMLR, 2019.

Felix Hausdorff. Summationsmethoden und momentfolgen. i. *Mathematische Zeitschrift*, 9 (1):74–109, 1921.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, pages 1731–1741, 2017.

Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Prateek Jain, Praneeth Netrapalli, Sham M Kakade, Rahul Kidambi, and Aaron Sidford. Parallelizing stochastic gradient descent for least squares regression: mini-batching, averaging, and model misspecification. *The Journal of Machine Learning Research*, 18(1): 8258–8299, 2017.

Stanisław Jastrzębski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in SGD. *arXiv preprint arXiv:1711.04623*, 2017.

Stanisław Jastrzębski, Zachary Kenton, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. On the relation between the sharpest directions of DNN loss and the SGD step length. 2018.

Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in neural information processing systems*, pages 586–594, 2016.

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. *arXiv preprint arXiv:1609.04836*, 2016.

Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

Frederik Kunstner, Lukas Balles, and Philipp Hennig. Limitations of the empirical Fisher approximation. *arXiv preprint arXiv:1905.12558*, 2019.

Harold Kushner and G George Yin. *Stochastic approximation and recursive algorithms and applications*, volume 35. Springer Science & Business Media, 2003.

Simon Lacoste-Julien, Mark Schmidt, and Francis Bach. A simpler approach to obtaining an o (1/t) convergence rate for the projected stochastic subgradient method. *arXiv preprint arXiv:1212.2002*, 2012.

Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*, 2020.

Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.

Yuanzhi Li, Colin Wei, and Tengyu Ma. Towards explaining the regularization effect of initial large learning rate in training neural networks. In *Advances in Neural Information Processing Systems*, pages 11669–11680, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Vladimir A Marčenko and Leonid Andreevich Pastur. Distribution of eigenvalues for some sets of random matrices. *Mathematics of the USSR-Sbornik*, 1(4):457, 1967.

James Martens. Deep learning via Hessian-free optimization. In *ICML*, volume 27, pages 735–742, 2010.

James Martens. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014.

James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417, 2015.

James Martens and Ilya Sutskever. Training deep and recurrent networks with Hessian-free optimization. In *Neural networks: Tricks of the trade*, pages 479–535. Springer, 2012.

Gérard Meurant and Zdeněk Strakoš. The Lanczos and conjugate gradient algorithms in finite precision arithmetic. *Acta Numerica*, 15:471–542, 2006.

Philipp Moritz, Robert Nishihara, and Michael Jordan. A linearly-convergent stochastic l-bfgs algorithm. In *Artificial Intelligence and Statistics*, pages 249–258. PMLR, 2016.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

Sean O'Rourke et al. A note on the marchenko-pastur law for a class of random matrices with dependent entries. *Electronic Communications in Probability*, 17, 2012.

Vardan Papyan. The full spectrum of deep net Hessians at scale: Dynamics with sample size. *arXiv preprint arXiv:1811.07062*, 2018.

Vardan Papyan. Traces of class/cross-class structure pervade deep learning spectra. *Journal of Machine Learning Research*, 21(252):1–64, 2020.

Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Barak A Pearlmutter. Fast exact multiplication by the Hessian. *Neural computation*, 6(1): 147–160, 1994.

Jeffrey Pennington and Yasaman Bahri. Geometry of neural network loss surfaces via random matrix theory. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2798–2806. JMLR. org, 2017.

Jeffrey Pennington and Pratik Worah. The spectrum of the fisher information matrix of a single-hidden-layer neural network. In *Advances in Neural Information Processing Systems*, pages 5410–5419, 2018.

Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. *arXiv preprint arXiv:1109.5647*, 2011.

Farbod Roosta-Khorasani and Uri Ascher. Improved bounds on sample size for implicit matrix trace estimators. *Foundations of Computational Mathematics*, 15(5):1187–1212, 2015.

Farbod Roosta-Khorasani and Michael W Mahoney. Sub-sampled newton methods ii: Local convergence rates. *arXiv preprint arXiv:1601.04738*, 2016.

Levent Sagun, Léon Bottou, and Yann LeCun. Eigenvalues of the Hessian in deep learning: Singularity and beyond. *arXiv preprint arXiv:1611.07476*, 2016.

Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the Hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

Christopher J Shallue, Jaehoon Lee, Joseph Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E Dahl. Measuring the effects of data parallelism on neural network training. *arXiv preprint arXiv:1811.03600*, 2018.

Ohad Shamir and Tong Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International conference on machine learning*, pages 71–79. PMLR, 2013.

Jack W Silverstein and ZD Bai. On the empirical distribution of eigenvalues of a class of large dimensional random matrices. *Journal of Multivariate analysis*, 54(2):175–192, 1995.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Samuel L Smith and Quoc V Le. A bayesian perspective on generalization and stochastic gradient descent. *arXiv preprint arXiv:1710.06451*, 2017.

Samuel L Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V Le. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

Charles Stein. A bound for the error in the normal approximation to the distribution of a sum of dependent random variables. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 2: Probability Theory*, pages 583–602, Berkeley, Calif., 1972. University of California Press.

Terence Tao. *Topics in random matrix theory*, volume 132. American Mathematical Soc., 2012.

Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Normalized flat minima: Exploring scale invariant definition of flat minima for neural networks using pac-bayesian analysis. In *International Conference on Machine Learning*, pages 9636–9647. PMLR, 2020.

Shashanka Ubaru, Jie Chen, and Yousef Saad. Fast estimation of tr(f(a)) via stochastic Lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1075–1099, 2017.

Dan V Voiculescu, Ken J Dykema, and Alexandru Nica. *Free random variables*. Number 1. American Mathematical Soc., 1992.

Ke Wang, Geoff Pleiss, Jacob Gardner, Stephen Tyree, Kilian Q Weinberger, and Andrew Gordon Wilson. Exact gaussian processes on a million data points. *Advances in Neural Information Processing Systems*, 32:14648–14659, 2019.

Lei Wu, Zhanxing Zhu, et al. Towards understanding generalization of deep learning: Perspective of loss landscapes. *arXiv preprint arXiv:1706.10239*, 2017.

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*, 31:8279–8288, 2018.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George Dahl, Chris Shallue, and Roger B Grosse. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. In *Advances in Neural Information Processing Systems*, pages 8194–8205, 2019.

Appendix A. Proof of the Central Lemma

A full proof of Theorem 4, which rests heavily on disparate yet known results in the literature (Götze et al., 2012; Benaych-Georges and Nadakuditi, 2011; Bun et al., 2017) would span many dozens of pages, repeating prior work. We hence adopt an alternative proof strategy, which we hope is understandable and relatable to a machine learning audience, for which this work is intended. We first introduce a minimum amount of necessary random matrix theory background. We then prove Theorem 4, but under the stronger assumptions that the elements of the fluctuation matrix are i.i.d. Gaussian (the Gaussian Orthogonal Ensemble (Tao, 2012)). To understand why this makes sense, we consider the key ingredients of the proof

- The fluctuation matrix converges to the semi-circle law (which we introduce and explain in the next Section);
- the spectral perturbation low-rank empirical Hessian by the fluctuation matrix can be computed analytically using perturbation theory;
- By Lemma 2, the scaling relationships which characterise the extent of the noise perturbation as a function of batch size can be analysed.

Hence the only difference between the simplified proof and Theorem 4, is that we have more general conditions for the convergence semicircle law, which are detailed extensively in Götze et al. (2012). The other two key components proceed in an identical fashion.

A.1 Background

Following the notation of (Bun et al., 2017) the resolvent of a matrix H is defined as

$$\boldsymbol{G}_H(z) = (z\boldsymbol{I}_N - \boldsymbol{H})^{-1} \tag{35}$$

with $z = x + i\eta \in \mathbb{C}$. The normalised trace operator of the resolvent, in the $N \to \infty$ limit

$$\mathcal{S}_N(z) = \frac{1}{N}\mathrm{Tr}[\boldsymbol{G}_H(z)] \xrightarrow{N\to\infty} \mathcal{S}(z) = \int \frac{\rho(\lambda)}{z-\lambda} du \tag{36}$$

is known as the Stieltjes transform of the eigenvalue density ρ. The functional inverse of the Siteltjes transform, is denoted the blue transform $\mathcal{B}(\mathcal{S}(z)) = z$. The $\mathcal{R}$ transform is thence defined as

$$\mathcal{R}(w) = \mathcal{B}(w) - \frac{1}{w} \tag{37}$$

The following definition formally defines a Wigner matrix:

Definition 8 *Let $\{Y_i\}$ and $\{Z_{ij}\}_{1\leq i\leq j}$ be two real-valued families of zero mean, i.i.d. random variables, Furthermore suppose that $\mathbb{E}|Z_{i,j}|^2 = 1$ and for each $k \in \mathbb{N}$*

$$\max(E|Z_{i,j}|^k, E|Y_1|^k) < \infty \tag{38}$$

Consider an $n \times n$ symmetric matrix $\boldsymbol{M}_n$, whose entries are given by

$$\begin{cases} \boldsymbol{M}_n(i,i) = Y_i \\ \boldsymbol{M}_n(i,j) = Z_{ij} = \boldsymbol{M}_n(j,i) \end{cases} \tag{39}$$

The Matrix $\boldsymbol{M}_n$ is known as a real symmetric Wigner matrix.

Theorem 9 *Let $\{\boldsymbol{M}_n\}_{n=1}^{\infty}$ be a sequence of Wigner matrices, and for each n denote $\boldsymbol{X}_n = \boldsymbol{M}_n/\sqrt{n}$. Then $\rho(\lambda)$, converges weakly, almost surely to the semicircle distribution,*

$$d\mu(\lambda) = \rho(\lambda)d\lambda = \frac{1}{2\pi}\sqrt{4-\lambda^2}\mathbf{1}_{|\lambda|\leq 2}\, d\lambda. \tag{40}$$

Crucially for our calculations, it is known that the $\mathcal{R}$ transform of the Wigner matrix $\boldsymbol{W}$ where the variance is σ^2 instead of 1 is given by:

$$\mathcal{R}_W(z) = \sigma^2 z. \tag{41}$$

Free random matrices: The property of freeness for non commutative random matrices can be considered analogously to the moment factorisation property of independent random variables. Let us denote the normalized trace operator, which is equal to the first moment of the spectral density

$$\psi(H) = \frac{1}{N}\mathrm{Tr}\boldsymbol{H} = \frac{1}{N}\sum_{i=1}^{N}\lambda_i = \int_{\lambda\in\mathcal{D}} d\mu(\lambda)\lambda \tag{42}$$

We say matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ for which $\psi(\boldsymbol{A}) = \psi(\boldsymbol{B}) = 0$ (We can always consider the transform $\boldsymbol{A} - \psi(\boldsymbol{A})\boldsymbol{I}$) are free if they satisfy for any integers $n_1..n_k$ with $k \in \mathbb{N}^+$

$$\psi(\boldsymbol{A}^{n_1}\boldsymbol{B}^{n_2}\boldsymbol{A}^{n_3}\boldsymbol{B}^{n_4}) = \psi(\boldsymbol{A}^{n_1})\psi(\boldsymbol{B}^{n_2})\psi(\boldsymbol{A}^{n_3})\psi(\boldsymbol{A}^{n_4}). \tag{43}$$

A.2 Proof of the Main Lemma

Recall that we wish to derive the values of the extremal eigenvalues of the matrix sum $\boldsymbol{M} = \boldsymbol{A} + \boldsymbol{\epsilon}(\boldsymbol{w})/\sqrt{P}$, where $\boldsymbol{\epsilon}(\boldsymbol{w})$ has a limiting spectral density given by the semicircle law. In this section we show by the definition of the Stieltjes transform (which has a one to one correspondence with the spectral density) that a finite rank perturbation of the Stieltjes transform (corresponding to the eigenvalues of the matrix $\boldsymbol{A}$) can be dealt with perturbation theory.

The Stijeles transform of the matrix $\boldsymbol{\epsilon}(\boldsymbol{w})$ with corresponding semicircle eigenvalue distribution can be written as (Tao, 2012)

$$\mathcal{S}_{\boldsymbol{\epsilon}}(z) = \frac{z \pm \sqrt{z^2 - 4\sigma_\epsilon^2}}{2\sigma_\epsilon^2}. \tag{44}$$

From the definition of the Blue transform, we hence have

$$\begin{aligned} z &= \frac{\mathcal{B}_{\boldsymbol{\epsilon}}(z) \pm \sqrt{\mathcal{B}_{\boldsymbol{\epsilon}}^2(z) - 4\sigma_\epsilon^2}}{2\sigma_\epsilon^2} \\ \mathcal{B}_{\boldsymbol{\epsilon}}(z) &= \frac{1}{z} + \sigma_\epsilon^2 z \\ \mathcal{R}_{\boldsymbol{\epsilon}}(z) &= \sigma_\epsilon^2 z. \end{aligned} \tag{45}$$

Computing the $\mathcal{R}$ transform of the rank 1 matrix $\boldsymbol{A}$ (which has a non-trivial eigenvalue $\lambda_1 > \sigma_\epsilon$) using the Stieltjes transform (Bun et al., 2017), we find the effect on the spectrum is given by:

$$\mathcal{S}_{\boldsymbol{A}}(u) = \frac{1}{N}\frac{1}{u-\lambda_1} + \left(1 - \frac{1}{N}\right)\frac{1}{u} = \frac{1}{u}\left[1 + \frac{1}{N}\frac{\lambda_1}{1-u^{-1}\lambda_1}\right] \tag{46}$$

We can use perturbation theory similar to in Equation (45) to find the Blue and $\mathcal{R}$ transform which to leading order gives

$$\begin{aligned} \mathcal{B}_{\boldsymbol{A}}(\omega) &= \frac{1}{w} + \frac{\lambda_1}{N(1-\omega\lambda_1)} + \mathcal{O}(N^{-2}) \\ \mathcal{R}_{\boldsymbol{A}}(\omega) &= \frac{\lambda_1}{N(1-\omega\lambda_1)} + \mathcal{O}(N^{-2}) \end{aligned} \tag{47}$$

Setting $\omega = \mathcal{S}_{\boldsymbol{M}}(z)$ so

$$z = \mathcal{B}_{\boldsymbol{A}}(\mathcal{S}_{\boldsymbol{M}}(z)) + \frac{\lambda_1}{N(1-\lambda_1\mathcal{S}_{\boldsymbol{M}}(z))} + \mathcal{O}(N^{-2}) \tag{48}$$

using the ansatz of $\mathcal{S}_{\boldsymbol{M}}(z) = \mathcal{S}_0(z) + \frac{\mathcal{S}_1(z)}{N} + \mathcal{O}(N^{-2})$ we find that $\mathcal{S}_0(z) = \mathcal{S}_{\epsilon(w)}(z)$ and using that $\mathcal{B}'_{\boldsymbol{\epsilon}}(\mathcal{S}_{\boldsymbol{\epsilon}})(z)) = \frac{1}{\mathcal{S}_{\boldsymbol{\epsilon}}(z)}$, we conclude that

$$\mathcal{S}_1(z) = -\frac{\lambda_1 \mathcal{S}'_{\epsilon(w)}(z)}{1 - \mathcal{S}_{\epsilon(w)}(z)\lambda_1} \tag{49}$$

and hence

$$\mathcal{S}_{\boldsymbol{M}}(z) \approx \mathcal{S}_{\epsilon(w)}(z) - \frac{1}{N}\frac{\lambda_1 \mathcal{S}'_{\epsilon(w)}(z)}{1 - \mathcal{S}_{\epsilon(w)}(z)\lambda_1} \tag{50}$$

In the large N limit the correction only survives if $\mathcal{S}_{\epsilon(w)}(z) = 1/\lambda_1$

$$\begin{aligned} \mathcal{S}_{\epsilon(w)}(z) &= \frac{1}{\lambda_1} \\ \frac{2\sigma_\epsilon^2}{\lambda_1} &= z \pm \sqrt{z^2 - 4\sigma_\epsilon^2} \\ \therefore z &= \lambda_1 + \frac{\sigma_\epsilon^2}{\lambda_1} \end{aligned} \tag{51}$$

The same proof also holds for $\lambda_P < -2\sigma_\epsilon$ and hence the second part of the Lemma also follows.

A.3 Overlap between Eigenvectors of the Batch and Empirical Hessian

Theorem 10 *The squared overlap $|\boldsymbol{\phi}_i' \boldsymbol{\phi}_i|^2$ between an eigenvector of the batch Hessian $\boldsymbol{\phi}_i'$ and that of the empirical Hessian $\boldsymbol{\phi}_i$ is given by*

$$|\boldsymbol{\phi}_i' \boldsymbol{\phi}_i|^2 = \left\{ \begin{array}{ll} 1 - \frac{P}{\mathfrak{b}} \frac{\sigma_\epsilon^2}{\lambda_1^2}, & \text{if } \lambda_1 > \sqrt{\frac{P}{\mathfrak{b}}} \sigma_\epsilon \\ 0, & \text{otherwise} \end{array} \right\} \tag{52}$$

For this theorem we utilise the following Lemma

Lemma 11 *Denote by ϕ_1' as the eigenvector associated with the largest eigenvalue of the matrix sum $\boldsymbol{M} = \boldsymbol{A} + \boldsymbol{\epsilon}(\boldsymbol{w})/\sqrt{P}$, where $\boldsymbol{A} \in \mathbb{R}^{P \times P}$ is a matrix of finite rank r with largest eigenvalue λ_1 and $\boldsymbol{\epsilon}(\boldsymbol{w}) \in \mathbb{R}^{P \times P}$ with limiting spectral density $p(\lambda)$ satisfying the semicircle law $p(\lambda) = \frac{\sqrt{4\sigma_\epsilon^2 - \lambda^2}}{2\pi\sigma_\epsilon^2}$. Then we have*

$$|\boldsymbol{\phi}_i' \boldsymbol{\phi}_i|^2 = \left\{ \begin{array}{ll} 1 - \frac{\sigma_\epsilon^2}{\lambda_1^2}, & \text{if } \lambda_1 > 2\sigma_\epsilon \\ 0, & \text{otherwise} \end{array} \right\} \tag{53}$$

The result and proof of this Lemma can be found in Benaych-Georges and Nadakuditi (2011). Then the proof of the main theorem proceeds identically to that of Theorem 4.

Appendix B. Non unit variance Marchenko-Pastur Stieltjes Transform

We now derive the Stieltjes transform of the generalised non-unit variance Marchenko-Pastur density. This derivation closely follows (Feier, 2012), but generalises the result. Original texts pertaining to the mathematics can be found in Marčenko and Pastur (1967); Silverstein and Bai (1995). Note that Feier (2012) use a different convention for the Stieltjes transform

$$\mathcal{S}_P(z) = \int_{\mathbb{R}} \frac{1}{x - z} \rho(x) dx = \frac{1}{P} \operatorname{Tr}(\boldsymbol{M}_n/\sqrt{P} - z\boldsymbol{I})^{-1} \tag{54}$$

We consider a series of matrices

$$\boldsymbol{X}_N = \left(r_i^s / \sqrt{P} \right)_{1 \leq i \leq P, 1 \leq s \leq N} \tag{55}$$

where the entries r_i^s are 0 mean and variance σ^2 The Wishart matrix $\boldsymbol{W}_P = \boldsymbol{X}_P \boldsymbol{X}_P^T$, where $(\boldsymbol{X}_P \boldsymbol{X}_P)_{i,j} = \frac{1}{N} \sum_{s=1}^N r_i^s r_j^s$.. Clearly, $\boldsymbol{W}_P$ can be written as the sum of rank-1 contributions $\boldsymbol{W}_P^s = (r_i^s r_j^s)_{1 \leq i,j \leq P}$. Now as each element is of mean 0 and variance σ^2, the expectation of the sum of the elements squared is given by $P^2\sigma^4/N^2 = Tr([\boldsymbol{W}_P^s]^2) = \lambda^2$ and hence the only eigenvalue (the contribution is rank 1) is given by $\lambda = \frac{P}{T}\sigma^2 = \beta\sigma^2$. For large P, by the weak law of large numbers, this is also true for a single realisation of $\boldsymbol{W}_P^s$. By the strong law of large numbers the column vectors $\boldsymbol{r}^s = [r_1^s ... r_P^s]^T$ and $\boldsymbol{r}^{s'}$ are almost surely orthogonal as $P \to \infty$ and hence the matrices $\boldsymbol{W}_P^s$ are asymptotically free (Voiculescu et al., 1992)

The Stieltjes transform $\mathcal{S}(z)$ of $\boldsymbol{W}_P^s$

$$\frac{1}{P} \operatorname{Tr}(\boldsymbol{W}_P^s - z\boldsymbol{I})^{-1} = -\frac{1}{P} \sum_{k=0}^{\infty} \frac{\operatorname{Tr}(\boldsymbol{W}_P^s)^k}{z^{k+1}} = -\frac{1}{P} \left(\frac{P-1}{z} + \frac{1}{z - \beta\sigma^2} \right) \tag{56}$$

Solving the quadratic for z, completing the square, dropping low order terms in P and noting by the definition of the Stieltjes transform that for large $|z| \sim -\frac{1}{z}$

$$z = \frac{P(s\beta\sigma^2 - 1) \pm \sqrt{P^2(s\beta\sigma^2 - 1)^2 + 4Ps(P-1)\beta\sigma^2}}{2Ps} = \frac{P(s\beta\sigma^2 - 1) \pm \sqrt{P^2(s\beta\sigma^2 + 1)^2 - 4Ps\beta\sigma^2}}{2Ps}$$
$$z \approx \frac{P(s\beta\sigma^2 - 1) - P(s\beta\sigma^2 + 1) - \frac{2Ps\beta\sigma^2}{s\beta\sigma^2+1}}{2Ps} = -\frac{1}{s} + \frac{\beta\sigma^2}{P(s\beta\sigma^2+1)} \tag{57}$$

Hence as the $\boldsymbol{W}_P$ is the free convolution (Voiculescu et al., 1992) of the random matrices $\boldsymbol{W}_P^s$ we simply multiply the $\mathcal{R}$ transform of each matrix by N and as $\beta = P/N$ so:

$$\begin{aligned} \mathcal{R}_{\boldsymbol{W}_P}(s) &= N \times \left(z - \frac{1}{s}\right) = \frac{N\beta\sigma^2}{P(s\beta\sigma^2+1)} = \frac{\sigma^2}{(s\beta\sigma^2+1)} \\ \mathcal{B}_{\boldsymbol{W}_P}(s) &= z = -\frac{1}{s} + \frac{\sigma^2}{(s\beta\sigma^2+1)} \end{aligned} \tag{58}$$

and hence

$$\mathcal{S}_{\boldsymbol{W}_P} = \frac{-(z + \sigma^2(1-\beta)) + \sqrt{(z+\sigma^2(1-\beta))^2 - 4\beta\sigma^2 z}}{2\beta\sigma^2 z} \tag{59}$$

From here, using the definition of the Stieltjes transform and the relationship to the spectral density, $\mathrm{Im}_{y\to 0}(\mathcal{S}_{\boldsymbol{W}_P}(x+iy))/2\pi i$ we have the celebrated generalised Marchenko-Pastur result.

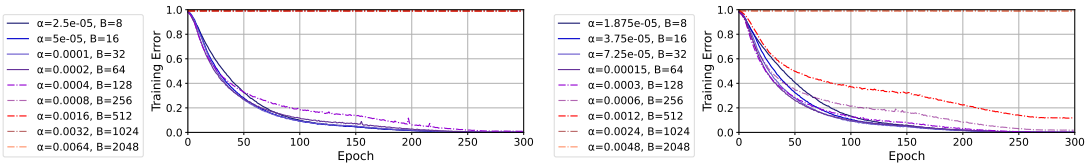
$$\rho(y) = \frac{\sqrt{4\beta\sigma^2 y - (y + \sigma^2(1-\beta))^2}}{2\beta\sigma^2 y} \tag{60}$$

Noting that the Stieltjes transform from Feier (2012) is reversed in the convention of the sign (Bun et al., 2017), we take $z \to -z$. Now we apply the T transform, given by $\mathcal{T}(z) = z\mathcal{S}(z) - 1$ and the result from Benaych-Georges and Nadakuditi (2011), i.e $\lambda_i' = \mathcal{T}(\frac{1}{\lambda_i})$, where the dash denotes the eigenvalue corresponding to the batch Hessian (instead of the empirical which is fixed).

$$\mathcal{T}(z) = \frac{z - \sigma^2(1+\beta) - \sqrt{(z+\sigma^2(1-\beta))^2 - 4\beta\sigma^2 z}}{2\beta\sigma^2}. \tag{61}$$

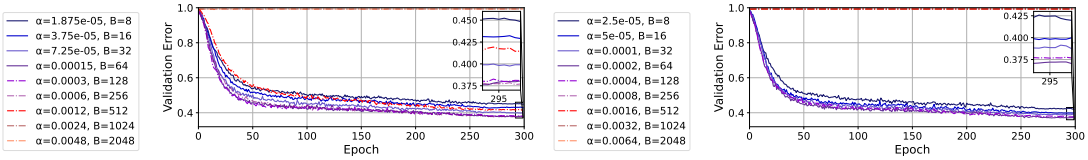
Appendix C. Differentiating Learning Rate Schedules

As shown in Figure 38, in the instances where training occurs, the training curves between slightly larger or smaller learning rates is indistinguishable. Note that the curves for $B = 128, 64$ crosses at seemingly identical points, despite the learning rate being 25%*smaller*. Clearly the training profile is different between curves that train and those that don't (when a too large a learning rate has been used and training never commences). However note that even for schedules with train indistinguishable training curves,the validation curve, shown in Figure 39 can differ significantly. Quite specifically for $B = 8, 16, 32$ there is an upwards shift in validation error of around 2%, for learning rates which have been decreased by 25%, despite indistinguishable training curves. We expect schedules parameterised by a different learning rates to traverse the non-convex loss surface in a different way. Specifically larger initial learning rate schedules would be expected to escape sharper local minima earlier in



(a) Training Error against Epoch (b) Training Error against Epoch

Figure 38: **Training Error trajectories do not well differentiate between learning rates if training occurs.** Training error of the VGG-16 architecture, without batch normalisation (BN) on CIFAR-100, with no weight decay $\gamma = 0$ and initial learning rate α_0



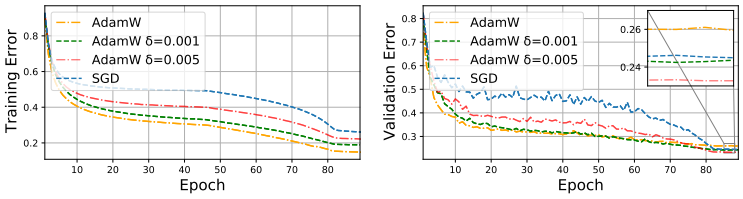
(a) Validation Error against Epoch (b) Validation Error against Epoch

Figure 39: **Validation Error trajectories distinguish between learning rates used.** Validation error of the VGG-16 architecture, without batch normalisation (BN) on CIFAR-100, with no weight decay $\gamma = 0$ and initial learning rate α_0.

training before the learning rate decay kicks in. Given that high capacity neural networks which are capable of easily memorising the data and hence there are many points of low training loss/error in the training risk, which may not be the result of following a similar trajectory in the loss surface, we consider the validation error plot (and final value) as more indicative to discriminate between effective scaling regimens.

C.1 Different Initailisations Give Similar Performance

We show here in 40c that for different seeds using SGD, Adam and Adam-δ where we simply tune the coefficient of the numerical stability coefficient to a larger value $\epsilon \to 10^{-4}$ instead of 10^{-8}, gives very consistent performance across a set of data sets and networks.



(a) ResNet-50 Training Error (b) ResNet-50 Testing Error

Data/Model	SGD	Adam-D	Adam
C100/VGG16	65.3 ± 0.6	65.5 ± 0.7	61.9 ± 0.4
ImgNet/Res50	75.7 ± 0.1	76.6 ± 0.1	-

(c) Statistical Significance

Figure 40: (a-b) The influence of δ on the generalisation gap. Train/Val curves for ResNet-50 on ImageNet. The generalisation gap is completely closed with an appropriate choice of δ. (c) Comparison of test accuracy across CIFAR 100 (5 seeds) and ImageNet (3 seeds). **Adam-D** denotes Adam with increased damping ($\delta = 5e^{-3}$ for CIFAR-100, $\delta = 1e^{-4}$ for ImageNet).

Appendix D. Lanczos algorithm

In order to empirically analyse properties of modern neural network spectra with tens of millions of parameters $N = \mathcal{O}(10^7)$, we use the Lanczos algorithm (Meurant and Strakoš, 2006), provided for deep learning by Granziol et al. (2019). It requires Hessian vector products, for which we use the *Pearlmutter trick* (Pearlmutter, 1994) with computational cost $\mathcal{O}(NP)$, where N is the data set size and P is the number of parameters. Hence for m steps the total computational complexity including re-orthogonalisation is $\mathcal{O}(NPm)$ and memory cost of $\mathcal{O}(Pm)$. In order to obtain accurate spectral density estimates we re-orthogonalise at every step (Meurant and Strakoš, 2006). We exploit the relationship between the Lanczos method and Gaussian quadrature, using random vectors to allow us to learn a discrete approximation of the spectral density. A quadrature rule is a relation of the form,

$$\int_a^b f(\lambda) d\mu(\lambda) = \sum_{j=1}^{M} \rho_j f(t_j) + R[f] \tag{62}$$

for a function f, such that its Riemann-Stieltjes integral and all the moments exist on the measure $d\mu(\lambda)$, on the interval $[a, b]$ and where $R[f]$ denotes the unknown remainder. The nodes t_j of the Gauss quadrature rule are given by the Ritz values and the weights (or mass) ρ_j by the squares of the first elements of the normalized eigenvectors of the Lanczos tridiagonal matrix (Golub and Meurant, 1994). The main properties of the Lanczos algorithm are summarized in the theorems 12,13

Theorem 12 *Let $H^{N\times N}$ be a symmetric matrix with eigenvalues $\lambda_1 \geq .. \geq \lambda_n$ and corresponding orthonormal eigenvectors $z_1, ..z_n$. If $\theta_1 \geq .. \geq \theta_m$ are the eigenvalues of the matrix T_m obtained after m Lanczos steps and $q_1, ...q_k$ the corresponding Ritz eigenvectors then*

$$\begin{aligned} \lambda_1 \geq \theta_1 \geq \lambda_1 - \frac{(\lambda_1 - \lambda_n)\tan^2(\theta_1)}{(c_{k-1}(1+2\rho_1))^2} \\ \lambda_n \leq \theta_k \leq \lambda_m + \frac{(\lambda_1 - \lambda_n)\tan^2(\theta_1)}{(c_{k-1}(1+2\rho_1))^2} \end{aligned} \tag{63}$$

where c_k is the Chebyshev polynomial of order k

Proof: see (Golub and Van Loan, 2012).

Theorem 13 *The eigenvalues of T_k are the nodes t_j of the Gauss quadrature rule, the weights w_j are the squares of the first elements of the normalized eigenvectors of T_k*

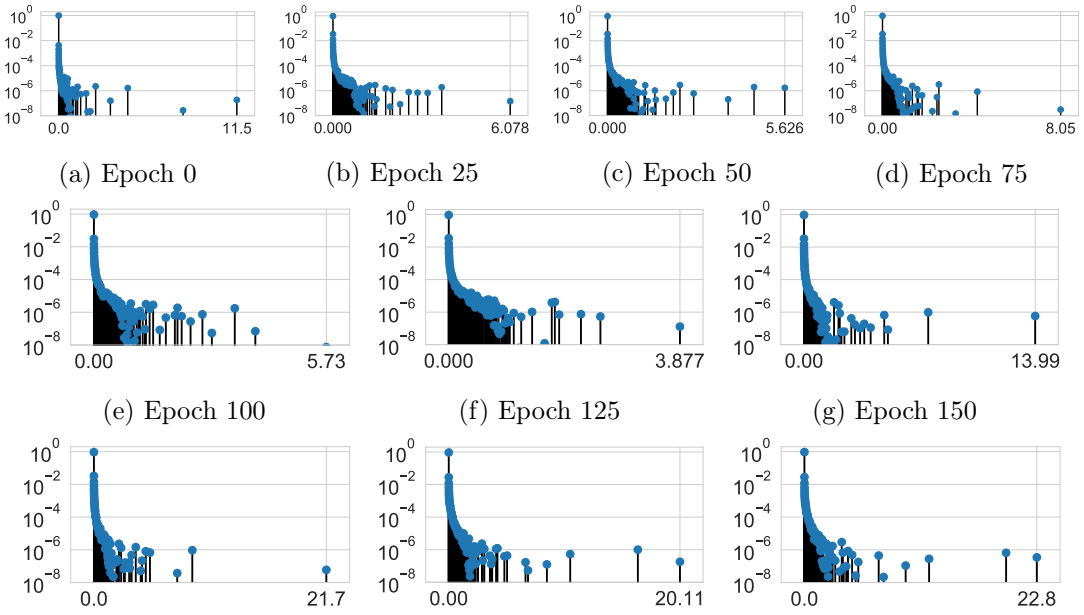
Proof: See (Golub and Meurant, 1994). The first term on the RHS of Equation 62 using Theorem 13 can be seen as a discrete approximation to the spectral density matching the first m moments $v^T H^m v$ (Golub and Meurant, 1994; Golub and Van Loan, 2012), where v is the initial seed vector. Using the expectation of quadratic forms, for zero mean, unit variance random vectors, using the linearity of trace and expectation

$$\mathbb{E}_v \mathrm{Tr}(v^T H^m v) = \mathrm{Tr}\mathbb{E}_v(vv^T H^m) = \mathrm{Tr}(H^m) = \sum_{i=1}^{N} \lambda_i = N \int_{\lambda \in \mathcal{D}} \lambda d\mu(\lambda) \tag{64}$$

The error between the expectation over the set of all zero mean, unit variance vectors v and the Monte Carlo sum used in practice can be bounded (Hutchinson, 1990; Roosta-Khorasani and Ascher, 2015). However in the high dimensional regime $N \to \infty$, we expect the squared overlap of each random vector with an eigenvector of H, $|v^T \phi_i|^2 \approx \frac{1}{N} \forall i$, with high probability. This result can be seen by computing the moments of the overlap between Rademacher vectors, containing elements $P(v_j = \pm 1) = 0.5$. Further analytical results for Gaussian vectors have been obtained (Cai et al., 2013).

Appendix E. Batch Normalisation Results

Given that the vast majority of image classification are run in conjunction with normalisation methods such as batch normalisation (Ioffe and Szegedy, 2015) and previous literature observing that batch normalisation suppresses outliers (Ghorbani et al., 2019) it is important to investigate whether the observations in terms of spectral structure and mini-batching effect are in any way invalidated with batch-normalisation. We hence present results on a variety of pre-activated residual networks. We show that the typical spectral density plots in the main text with well separated outliers, a large rank degeneracy and large increase in spectral width with mini-batching are visible also in batch normalised Resnets more commonly used in deep learning for both types of batch normalisation mode (explained in the next paragraph).



(a) Epoch 0 (b) Epoch 25 (c) Epoch 50 (d) Epoch 75

(e) Epoch 100 (f) Epoch 125 (g) Epoch 150

(h) Epoch 175 (i) Epoch 200 (j) Epoch 225

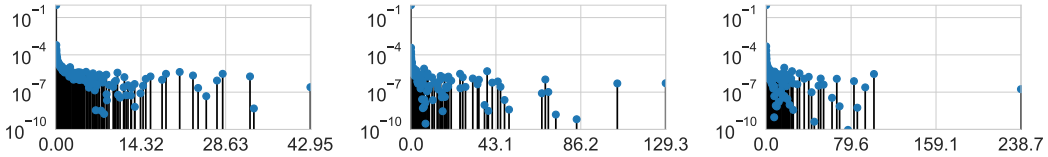
Figure 41: Generalised Gauss-Newton matrix full empirical spectrum for the PreResNet-110 on the CIFAR-100 data set, total training 225 epochs, batch norm train mode

Technical point on batch norm: Batch-normalisation, as alluded to in the main text function differently during training and during evaluation. When evaluating curvature, we thus have the option of choosing the setting of this functionality. We denote the same

properties as during training as batch norm train mode and those during evaluation as batch norm evaluation mode.We look at the Generalised Gauss-Newton matrix and Hessian of the PreResNet-110 in batch norm evaluation and training mode.

E.1 Generalised Gauss-Newton matrix - batch normalisation train mode

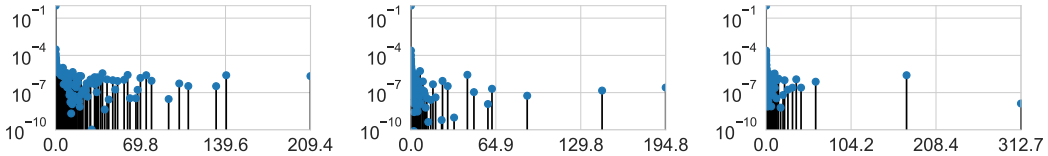
To show similarly that the Generalised Gauss-Newton matrix experiences severe spectral broadening when mini-batching, we take the same points in weight space as in Figure 41 but instead take stochastic samples of size $B = 128$, although the results are stochastic, they are stochastic around a significantly broadened spectrum, with some samples shown in Figure 42, for comparison. Where we see significant broadening.



(a) Epoch 175, $B = 128$ (b) Epoch 200, $B = 128$ (c) Epoch 225, $B = 128$

Figure 42: Generalised Gauss-Newton matrix full empirical spectrum for the PreResNet-110 on the CIFAR-100 data set, total training 225 epochs, batch norm train mode, samples taken with a batch of $B = 128$

E.2 Generalised Gauss-Newton matrix - Evaluation Mode



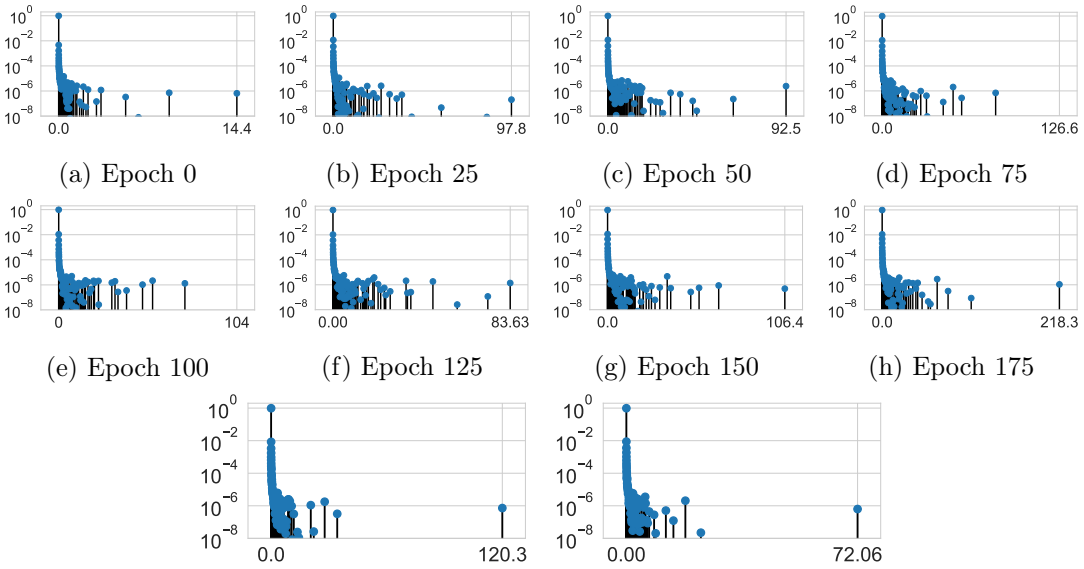
(a) Epoch 175, $B = 128$ (b) Epoch 200, $B = 128$ (c) Epoch 225, $B = 128$

Figure 43: Generalised Gauss-Newton matrix full empirical spectrum for the PreResNet-110 on the CIFAR-100 data set, total training 225 epochs, batch norm eval mode, $B = 128$ sub-sampled spectrum

Similarly to the previous section, we show in Figure 43 that even with batch normalisation in evaluation mode, the sub-sampling procedure induces extreme spectral broadening, compared to the same points in weight space with the full data set, shown in Figure 44. Again although the results are stochastic, with large variance the trend is consistent.

E.3 Hessian - Batch Normalisation Train Mode

Similar to the Generalised Gauss-Newton matrix, the Hessian has well separated both negative and positive outliers from the spectral bulk and a large rank degeneracy, these observations are consistent throughout training.

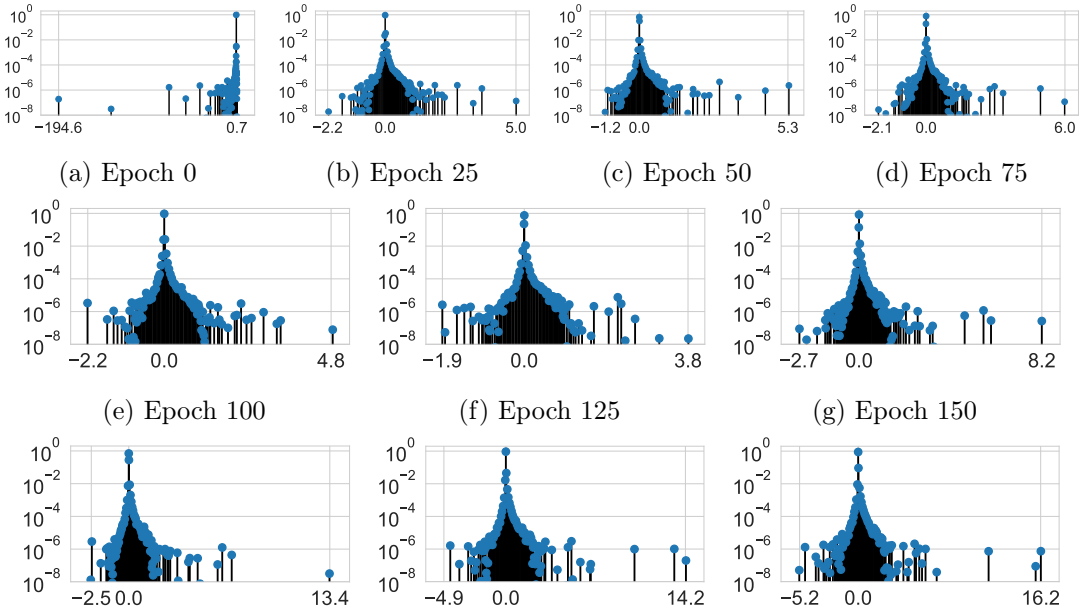


(a) Epoch 0 (b) Epoch 25 (c) Epoch 50 (d) Epoch 75

(e) Epoch 100 (f) Epoch 125 (g) Epoch 150 (h) Epoch 175

(i) Epoch 200 (j) Epoch 225

Figure 44: Generalised Gauss-Newton matrix full empirical spectrum for the PreResNet-110 on the CIFAR-100 data set, total training 225 epochs, batch norm eval mode



(a) Epoch 0 (b) Epoch 25 (c) Epoch 50 (d) Epoch 75

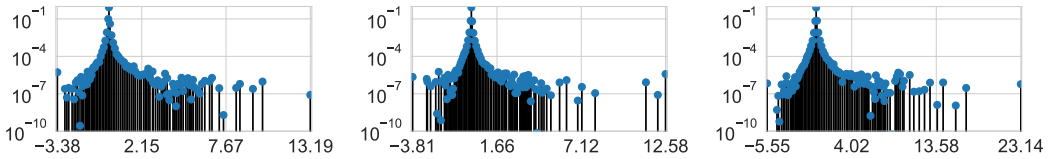
(e) Epoch 100 (f) Epoch 125 (g) Epoch 150

(h) Epoch 175 (i) Epoch 200 (j) Epoch 225

Figure 45: Hessian full empirical spectrum for the PreResNet-110 on the CIFAR-100 data set, total training 225 epochs, batch norm train mode

Similarly at all points in training, stochastic batch Hessians are shown to be significantly broadened, we see this by comparing the full data empirical Hessian spectrum 45 compared

to the Hessian at the same point in weight space but using only a batch size of $B = 128$ in Figure 46.

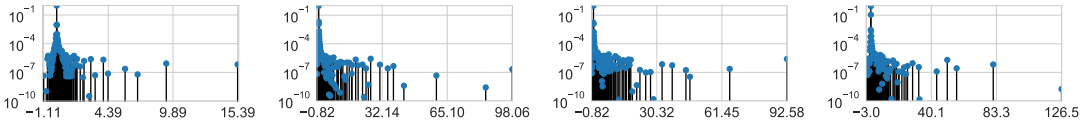


(a) Epoch 100, $B = 128$ (b) Epoch 125, $B = 128$ (c) Epoch 150, $B = 128$

Figure 46: Hessian batch spectrum for the PreResNet-110 on the CIFAR-100 data set, total training 225 epochs, batch norm train mode, $B = 128$

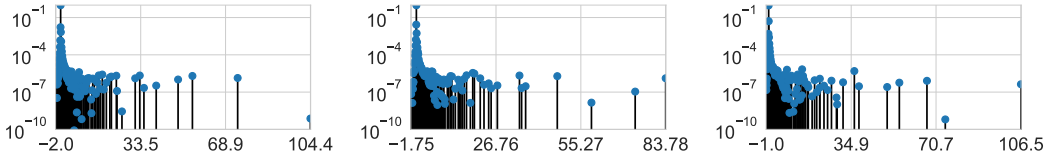
E.4 Hessian - Batch Normalisation Evaluation Mode

Similarly at all points in training, stochastic batch Hessians in evaluation mode are shown to be significantly broadened, we see this by comparing the full data empirical Hessian spectrum 48 compared to the Hessian at the same point in weight space but using only a batch size of $B = 128$ in Figure 49.



(a) Epoch 0 (b) Epoch 25 (c) Epoch 50 (d) Epoch 75

Figure 47: Hessian full empirical spectrum for the PreResNet-110 on the CIFAR-100 data set, total training 225 epochs, batch norm evaluation mode

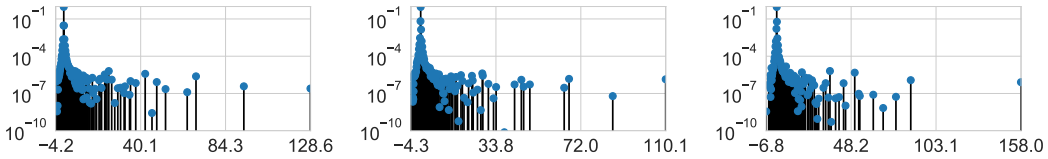


(a) Epoch 100 (b) Epoch 125 (c) Epoch 150

Figure 48: Hessian full empirical spectrum for the PreResNet-110 on the CIFAR-100 data set, total training 225 epochs, batch norm evaluation mode

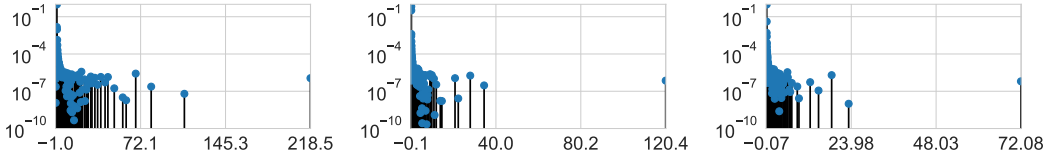
Appendix F. Alternative learning rate schedules and initialisation distance importance

One implicit assumption in Section 10 is that the largest learning rate which trains stably gives the best result. This informs our work as to how we should scale this rate as the batch size is increased. However it makes sense to consider how alternative more conservative



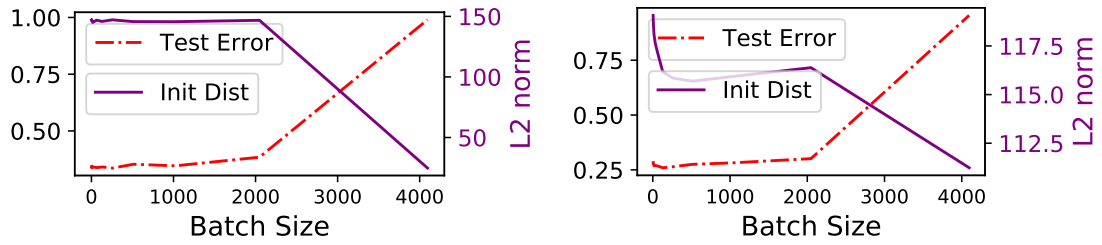
(a) Epoch 100, $B = 128$ (b) Epoch 125, $B = 128$ (c) Epoch 150, $B = 128$

Figure 49: Hessian batch spectrum for the PreResNet-110 on the CIFAR-100 data set, total training 225 epochs, batch norm train mode, $B = 128$



(a) Epoch 175 (b) Epoch 200 (c) Epoch 225

Figure 50: Hessian full empirical spectrum for the PreResNet-110 on the CIFAR-100 data set, total training 225 epochs, batch norm evaluation mode



(a) VGG-16 (b) VGG-16BN

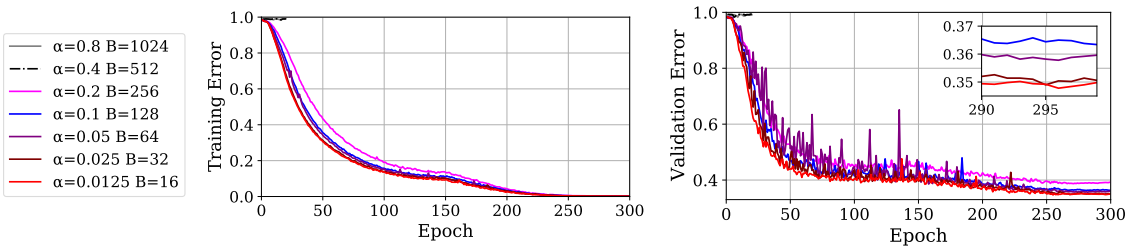
Figure 51: Test error as a function of initialisation distance for both the VGG-16 and VGG-16BN, for the CIFAR-100 data set. Learning rate is scaled linearly with batch size.

scaling rules might fare and whether they impact performance. In this section we also consider whether increased distance from Initialisation, as posited in Hoffer et al. (2017) is relevant for generalisation.We do this for the VGG-16 on the CIFAR-100 data set. Against a baseline validation accuracy of 65.82% for $B = 128$. For the $B = 1024$ case, our theoretically justified linearly increased learning rate of 0.08 gives an accuracy of 64.35%, whereas using the square root rule (Hoffer et al., 2017) suggestion of 0.028 only gives 61.08%, we note from Figure 6 that there are many well separated outliers. On the held out test set, the linear scaling solution has an error of 34.64% and a distance of 145.76 in $L2$ norm from the initialisation, whereas the square root scaled solution has an error of 37.55% and a distance of 67.44 from initialization. This indicates that as argued in Hoffer et al. (2017) that distance from initialisation seems to play an important role for generalisation. We test this further by looking at the initialisation distance across the set of similar test performing solutions for a constant learning rate to batch size ratio. Interestingly for the VGG-16 without batch

normalisation as shown in Figure 7a there is a strong link between initialisation distance in $L2$ norm and the test error. This relationship is much weaker and much smaller in magnitude when batch normalisation is utilised, as shown in Figure 7b. We even see the initialisation distance increasing as test error also increases.

Appendix G. SGD without momentum

We test the validity of our learning rate scaling for SGD (which is linear) for the case of momentum $\rho = 0$. We do not investigate the same for Adam because a $\beta_1, \beta_2 = 0.9, 0.99$ respectively and is typically not used at a value of zero. We find that for simply rescaling the learning rate by a factor of $\frac{1}{1-\rho}$ we have very similar results to SGD with momentum, which we show in Figure 52. We find interestingly that there seems to be significantly more stochasticity in the validation curves in the lower batch size regime than with a typical momentum of $\rho = 0.9$.



(a) Training Error against Epoch

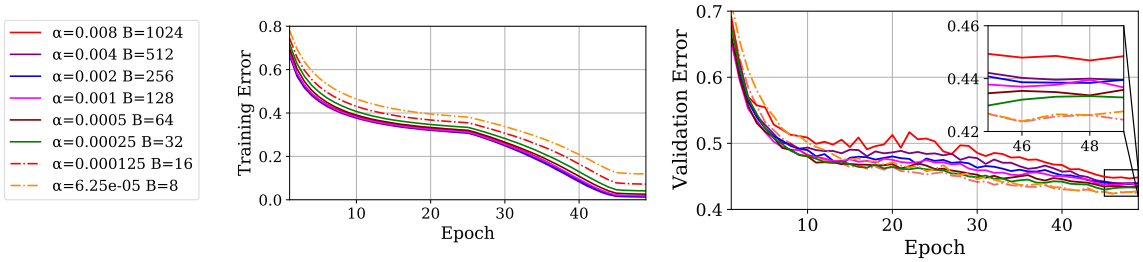
(b) Validation Error against Epoch

Figure 52: **Linear scaling is consistent up to a threshold.** Training and Validation error of the VGG-16 architecture, without batch normalisation (BN) on CIFAR-100, with no weight decay $\gamma = 0$ and initial learning rate $\alpha_0 = \frac{0.1B}{128}$ and no momentum

Appendix H. Limitations of the Square root (& Linear) scaling rules

We extend our observation space to include the ImageNet data set (Krizhevsky, 2014). In order to keep the multiple runs computationally feasible we use the downsampled 32×32 version (Chrabaszcz et al., 2017). This slightly increases the complexity of the task (there is less information in the image). We find that the 16 layer VGG (Simonyan and Zisserman, 2014), despite being originally designed for the larger version of this data set, to perform very badly - both in training and in test. Indeed, the results are so poor, indicating that the choice of architecture is inappropriate for this data set, that we instead use the 28×10 Wide-Residual network (Zagoruyko and Komodakis, 2016), with batch normalisation (Ioffe and Szegedy, 2015) and decoupled weight decay (Loshchilov and Hutter, 2017). Whilst this combination of "tricks" such as residual connections, batch normalisation and decoupled weight decay, all add extra layers of complexity to the loss landscape analysis that we do not consider in depth in this work, it brings us closer to a typical high performance training regimen. Note for example that typical uses of regularisation, such as decoupled weight decay are learning rate dependent.

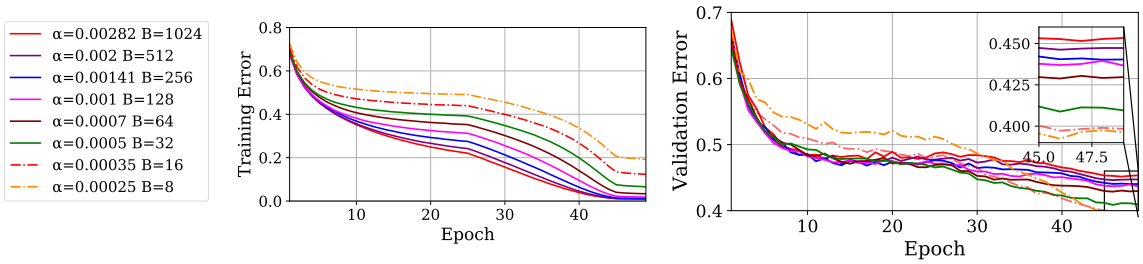
Complexities of the WideResNet with ImageNet32: Due to the large data set size (over $1m$ images), the computational complexity of running even a 50 epoch schedule on 1GPU is on the order of 2 days. This in conjunction with the fact that Residual networks in combination with Batch Normalisation are very stable to a wide variety of learning rates and can even converge after being run at very high learning rates at which their is initially no or very little training, makes settling on a "best" learning rate very difficult. This follows because each full run is very expensive and every run needs to go to completion to have a handle as to its performance. We hence follow (Granziol et al., 2020) and keep the default



(a) Training Error against Epoch

(b) Validation Error against Epoch

Figure 53: **Linear scaling rate does not hold for Adam.** Training and Validation error of the WideResNet-28 $\times$ 10 architecture, without batch normalisation (BN) on ImageNet-32, with no weight decay $\gamma = 0$ and initial learning rate $\alpha_0 = \frac{0.001B}{128}$, which varies as a function of batch size B.



(a) Training Error against Epoch

(b) Validation Error against Epoch

Figure 54: **Linear scaling rate does not hold for Adam.** Training and Validation error of the WideResNet-28 $\times$ 10 architecture, without batch normalisation (BN) on ImageNet-32, with no weight decay $\gamma = 0$ and initial learning rate $\alpha_0 = \frac{0.001\sqrt{B}}{\sqrt{128}}$, which varies as a function of batch size B.

learning rate of 0.001 for Adam with a decoupled weight decay of 0.01. Whilst we experiment with increased 0.003 and decreased 0.0003 learning rates respectively, we find the validation error on these runs to be worse, hence we consider 0.001 as our base "optimal" rate. We then use the square root and linear learning rate prescriptions which we respectively plot in Figures 53 and 54 respectively.

We find that both prescriptions train well. Interestingly, while in both prescriptions, reducing the batch size increases the training error (whilst decreasing the validation error)

we find the effect to be more pronounced with the square root rule compared to the linear regime. This indicates that this experimental setup follows a more aggressive than linear scaling rule, i.e. $\alpha \propto B^{1+\eta}$, where $\eta > 0$.

We note that for the validation error trajectories, which we have already discussed in Section 9 can serve as a proxy to movement in the loss surface, shows very similar trajectories for both rules until a batch size of 16 and 512, but starts to break down very quickly for a batch size of 1024 for the linear scaling case and not for the square root case. Results at both increased and decreased batch size are improved using the square root scaling rule derived in this paper than than the typically employed linear rule. We note that for both the linear and the square-root scalings that the curves with a batch size of 16&32 show significant deviation in trajectory from their larger batch counterparts. We see slower training, to a higher training error and a slower validation improvement, falling to a lower validation error in the end. This indicates an undiscovered relationship between batch normalisation, weight decay, residual connections and adaptive optimisation, which could be the subject of future research.