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

Diego Granziol
Machine Learning Research Group
University of Oxford
diego@robots.ox.ac.uk

Stefan Zohren
Machine Learning Research Group and Oxford-Man Institute for Quantitative Finance
University of Oxford
zohren@robots.ox.ac.uk

Stephen Roberts
Machine Learning Research Group and Oxford-Man Institute for Quantitative Finance
University of Oxford
sjrob@robots.ox.ac.uk

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 optimisation schemes for both stochastic gradient descent (linear scaling) and adaptive algorithms such as Adam (square root scaling). 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. For stochastic Second-order methods and adaptive methods, we derive that the minimal damping coefficient is proportional to the ratio of the learning rate to batch size. We validate our claims on the VGG/WideResNet architectures on the CIFAR-100 and ImageNet datasets.

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. 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 dataset 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;
- When the eigenvalues of the full dataset 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 dataset;
- This rigorous theoretical result 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, the setting here is much more general and less restrictive than in previous work;
- The minimum damping term of stochastic Second-order methods, often grid-searched as an extra hyper-parameter (Dauphin et al., 2014) or adjusted during training (Martens, 2016), is inversely proportional to the batch size when large learning rates are used;
- 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.
- 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 to back up this assertion.

The paper is structured as follows. The relevance of our work, key contributions and relationships to prior literature is detailed in Section 2. Section 3 details the random matrix theory framework modelling the noise due to mini-batching – it states the assumptions, lemmas and proofs. Section 4.2 extends the framework from Section 3 to strictly positive-definite matrices such as the Generalised Gauss-Newton matrix. Section 5 (which may be of greatest interest to deep learning practitioners) provides experimental validation for the theoretical claims, along with experiments to verify the key practical implications. Section 6 discusses several applications of the here presented theoretical results. Finally, we conclude in Section 7. 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 dataset, we can evaluate this difference in spectrum analytically. In this paper we develop this idea with two different assumptions corresponding to additive and multiplicative noise processes. 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 datasets (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.

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. Proofs of convergence for this class of methods explicitly require similarity between the sub-sampled and full dataset Hessian spectrum (Roosta-Khorasani and Mahoney, 2016). Hence, understanding the spectral perturbations due to mini-batching is of great importance for Second-order methods. We further derive an inverse relationship between the damping coefficient of stochastic Second-order methods and batch size, which is to our knowledge novel. Given that the damping coefficient is often extensively grid searched (Dauphin et al., 2014) or adapted during training (Martens, 2014), such a result may be helpful in the utilisation of Second-order methods.

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.

2.2 Related Work

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

Some of the ideas in this work are inspired by earlier unfinished work on the true loss surface (Granziol et al., 2018). This work has several shortcomings however, one being the focus on the true loss surface, which is of little practical relevance. A discussion highlighting further differences is given in Appendix D.

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 dataset 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 $B \ll N$) 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.

$$\boldsymbol{H}_{batch}(\boldsymbol{w}) = \boldsymbol{H}_{emp}.(\boldsymbol{w}) + \boldsymbol{\epsilon}(\boldsymbol{w})^1 \tag{4}$$

Rewriting the fluctuation matrix as $\boldsymbol{\epsilon}(\boldsymbol{w}) \equiv \boldsymbol{H}_{batch}(\boldsymbol{w}) - \boldsymbol{H}_{emp}(\boldsymbol{w})$ and assuming the mini-batch to be drawn independently from the dataset, we can infer

1. 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.

$$\begin{aligned}
\boldsymbol{\epsilon}(\boldsymbol{w}) &= \left(\frac{1}{B} - \frac{1}{N}\right) \sum_{j=1}^{B} \nabla^2 \ell(\boldsymbol{x}_j, \boldsymbol{w}; \boldsymbol{y}_j) - \frac{1}{N} \sum_{i=B+1}^{N} \nabla^2 \ell(\boldsymbol{x}_i, \boldsymbol{w}; \boldsymbol{y}_i) \\
\text{thus } \mathbb{E}(\boldsymbol{\epsilon}(\boldsymbol{w})_{j,k}) &= 0 \text{ and } \mathbb{E}(\boldsymbol{\epsilon}(\boldsymbol{w})_{j,k})^2 = \left(\frac{1}{B} - \frac{1}{N}\right) \mathrm{Var}[\nabla^2 \ell(\boldsymbol{x}, \boldsymbol{w}; \boldsymbol{y})_{j,k}].
\end{aligned} \tag{5}$$

Where B is the batch size and N the total dataset size. The expectation is taken with respect to the data generating distribution $\psi(\boldsymbol{x}, \boldsymbol{y})$. In order for the variance in Equation 5 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 a constant.*

Proof As the gradient of the empirical risk is L Lipschitz continous, as the empirical risk a sum over the samples, the gradient of the batch risk is also Lipschitz continous. 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 fluctuation matrix $\boldsymbol{\epsilon}(\boldsymbol{w})$ must be smaller than L. Hence using the Frobenius norm we can upper bound the matrix elements of $\boldsymbol{\epsilon}(\boldsymbol{w})$

$$\begin{aligned}
\mathrm{Tr}(\boldsymbol{\epsilon}(\boldsymbol{w})^2) &= \sum_{j,k=1}^{P} \boldsymbol{\epsilon}(\boldsymbol{w})_{j,k}^2 = \boldsymbol{\epsilon}(\boldsymbol{w})_{j=j',k=k'}^2 + \sum_{j \neq j', k \neq k'}^{P} \boldsymbol{\epsilon}(\boldsymbol{w})_{j,k}^2 = \sum_{i=1}^{P} \lambda_i^2 \\
\text{thus } \boldsymbol{\epsilon}(\boldsymbol{w})_{j=j',k=k'}^2 &\leq \sum_{i=1}^{P} \lambda_i^2 \leq PL^2 \text{ and } -\sqrt{P}L \leq \boldsymbol{\epsilon}(\boldsymbol{w})_{j=j',k=k'} \leq \sqrt{P}L.
\end{aligned} \tag{6}$$

■

As the domain of the Hessian elements under the data generating distribution is bounded, the moments of Equation 5 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), $(\frac{1}{B} - \frac{1}{N})^{-1/2}[\nabla^2 R_{true}(\boldsymbol{w}) - \nabla^2 R_{emp}(\boldsymbol{w})]_{jk} \xrightarrow[a.s]{} \mathcal{N}(0, \sigma_{jk}^2)$. ■

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 $P(\frac{1}{B} - \frac{1}{N}) = q > 0$. 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

$$\mathfrak{F}^{(i,j)} := \sigma\{\boldsymbol{\epsilon}(\boldsymbol{w})_{kl} : 1 \leq k \leq l \leq P, (k,l) \neq (i,j)\}, \quad q \leq i \leq j \leq P \tag{7}$$

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 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 spectra density $p(\lambda)$ of $\boldsymbol{\epsilon}(\boldsymbol{w}) \in \mathbb{R}^{P\times P}$ satisfies the semi circle 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 $\mathbb{E}(\boldsymbol{\epsilon}(\boldsymbol{w})_{i,j}^2|\mathfrak{F}^{i,j}) \neq \mathbb{E}(\boldsymbol{\epsilon}(\boldsymbol{w})_{i,j}^2) = \sigma_{i,j}^2$.

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. ■

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 $9i)$ 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.

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*

$$\lambda_1' = \left\{ \begin{array}{ll} \lambda_1 + \frac{P}{\mathfrak{b}}\frac{\sigma_\epsilon^2}{\lambda_1}, & \text{if } \lambda_1 > \sqrt{\frac{P}{\mathfrak{b}}}\sigma_\epsilon \\ 2\sqrt{\frac{P}{\mathfrak{b}}}\sigma_\epsilon, & \text{otherwise} \end{array} \right\}, \lambda_P' = \left\{ \begin{array}{ll} \lambda_P + \frac{P}{\mathfrak{b}}\frac{\sigma_\epsilon^2}{\lambda_P}, & \text{if } \lambda_P < -\sqrt{\frac{P}{\mathfrak{b}}}\sigma_\epsilon \\ -2\sqrt{\frac{P}{\mathfrak{b}}}\sigma_\epsilon, & \text{otherwise} \end{array} \right\}. \quad (8)$$

where $[\lambda_1, \lambda_P]$ are the extremal eigenvalues of $\boldsymbol{H}_{emp}(\boldsymbol{w})$, $\mathfrak{b} = B/(1-B/N)$ and B is the batch-size.[2] *Recall that σ_ϵ is defined in Theorem 3, through the limiting spectral density $p(\lambda)$ of $\boldsymbol{\epsilon}(\boldsymbol{w})$.*

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.

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

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 semi circle law* $p(\lambda) = \frac{\sqrt{4\sigma_\epsilon^2 - \lambda^2}}{2\pi\sigma_\epsilon^2}$. *Then we have*

$$\lambda'_1 = \left\{ \begin{array}{ll} \lambda_1 + \frac{\sigma_\epsilon^2}{\lambda_1}, & \textit{if } \lambda_1 > \sigma_\epsilon \\ 2\sigma_\epsilon, & \textit{otherwise} \end{array} \right\}, \lambda'_P = \left\{ \begin{array}{ll} \lambda_P + \frac{\sigma_\epsilon^2}{\lambda_P}, & \textit{if } \lambda_P < -\sigma_\epsilon \\ -2\sigma_\epsilon, & \textit{otherwise} \end{array} \right\}. \tag{9}$$

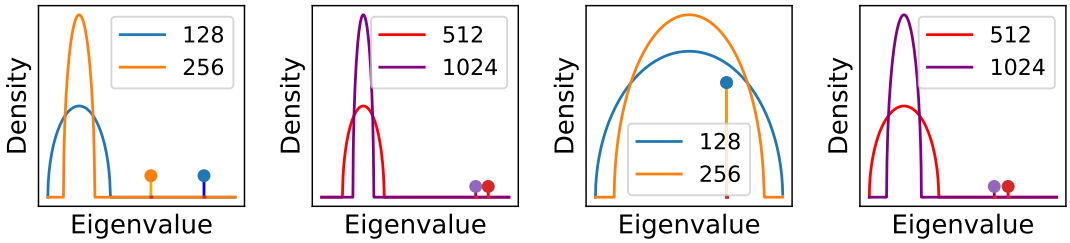
We now proceed with the proof of Theorem 4:

Proof The variance per element is a function of the batch size B and the size of the empirical dataset 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 Appendix C 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.

4.1 Illustration of the Key Result

We illustrate our key result (formalised in Theorem 4) in Figure 1.



(a) Linear scaling (b) Threshold (c) Square root scaling (d) Threshold

Figure 1: Variation of the spectral norm with batch size. The continuous region (bulk) corresponds to the fluctuation matrix induced by mini-batching (shown as a Semi Circle, whose width depends on the batch size) and the largest eigenvalue of the full Hessian is shown as a single peak. There are 3 major **scaling** regimes, which define how the largest eigenvalue decreases as the batch size increases. (a) If the largest eigenvalue is well seperated from the spectrum of the fluctuation matrix, *the scaling is linear until a threshold* (b) beyond which there is no major change.
(c) If the largest eigenvalue is smaller than the largest bulk eigenvalue, the scaling is proportional to the square root, until a threshold (d).
Crucially for deep learning optimisation, the scaling determines the allowed increase in learning rate

If the largest Hessian eigenvalue is well separated from the fluctuation matrix (continuous spectral

density), as shown in Figure 2a, then increasing the batch size, which reduces the spectral width of the fluctuation matrix, will have an approximately linear effect in reducing the spectral norm. This will hold up until a threshold, shown in Figure 1b, after which the spectral norm no longer appreciably changes in size. In the case that the spectral norm of the full dataset Hessian is smaller than that of the fluctuation matrix, shown in Figure 1c, the largest eigenvalue of the batch Hessian, which is given by the fluctuation matrix, will reduce as the square root of the batch size, again up until a critical level shown in Figure 1d. As discussed in Section 5 the allowed increase in learning rate as a function of batch size is proportional to the scaling.

4.2 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. This means that instead of having an additive noise model (where each element of the batch Hessian is perturbed by some additive noise), we instead consider a multiplicative noise model as in Bun et al. (2016).
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 and the multiplicative noise model.

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 batch Hessian may be expressed in terms of the activation σ at the output of the final layer $f(\boldsymbol{x})$ using the chain rule as

$$\boldsymbol{H}_{batch}(\boldsymbol{w})_{ij} = \sum_{b=1}^{B} \left(\sum_{k=0}^{d_y} \sum_{l=0}^{d_y} \frac{\partial^2 \sigma(f(\boldsymbol{x}))}{\partial f_l(\boldsymbol{x}) \partial f_k(\boldsymbol{x})} \frac{\partial f_l(\boldsymbol{x})}{\partial x_j} \frac{\partial f_k(\boldsymbol{x})}{\partial x_i} + \sum_{k=0}^{d_y} \frac{\partial \sigma(f(\boldsymbol{x}))}{\partial x_k} \frac{\partial^2 f_k(\boldsymbol{x})}{\partial x_j \partial x_i} \right) / B. \tag{10}$$

The first term on the RHS of Equation 10 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 $f(\boldsymbol{x})$ we rewrite the GGN matrix per sample as

$$\sum_{k,l=0}^{d_y} \sqrt{\frac{\partial^2 \sigma(f(\boldsymbol{x}))}{\partial f_l(\boldsymbol{x}) \partial f_k(\boldsymbol{x})}} \frac{\partial f_l(\boldsymbol{x})}{\partial x_j} \times \sqrt{\frac{\partial^2 \sigma(f(\boldsymbol{x}))}{\partial f_l(\boldsymbol{x}) \partial f_k(\boldsymbol{x})}} \frac{\partial f_k(\boldsymbol{x})}{\partial x_i} = \boldsymbol{J}_* \boldsymbol{J}_*^T, \tag{11}$$

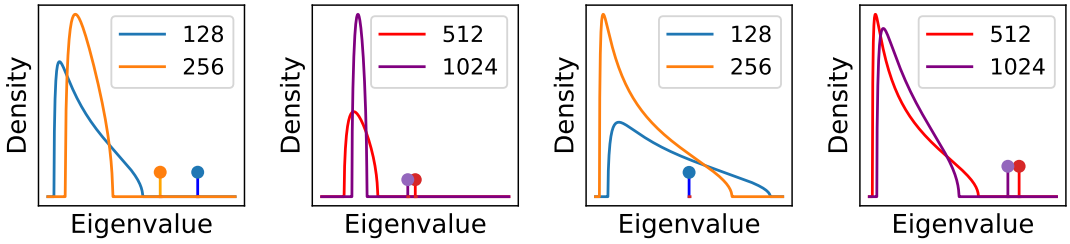
where we define $\boldsymbol{J}_*$ in order to retain a similarity for the GGN matrix in the case of the squared loss function (Pennington and Bahri, 2017), which has the form $\boldsymbol{G}(\boldsymbol{w}) = \boldsymbol{J}\boldsymbol{J}^T$. There are many potential candidate noise models, such as the free multiplicative and information plus noise model (Bun et al., 2016; Hachem et al., 2013), Typically for equations of the form in Equation 11, we would write $\boldsymbol{J}_{batch} = \boldsymbol{J}_{true}\boldsymbol{\epsilon}$ and hence $\boldsymbol{J}_{batch}\boldsymbol{J}_{batch}^T = \boldsymbol{J}_{true}\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T\boldsymbol{J}_{true}$ (Bun et al., 2017). The corresponding analysis, gives a very similar result to Theorem 4. The key take away 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 to be a low-rank perturbation of that fluctuation matrix.* For completeness, we derive the non unit variance Stieltjes transform of the Marchenko-Pastur

distribution in Appendix B. Following through the algebra, with some simple cancellations, we arrive at a qualitatively similar result to Theorem 4 for the perturbation of the batch GGN to its full dataset counterpart.

Theorem 6 *The extremal eigenvalue λ_1' of the matrix $\boldsymbol{G}_{batch}$, where $\boldsymbol{G}_{emp}$ has extremal eigenvalue λ_1, is given by*

$$\lambda_1' = \left\{ \begin{array}{ll} \frac{P\sigma^2}{\mathfrak{b}}(1+\frac{\beta}{\lambda_1})(1+\lambda_1), & \text{if } \lambda_1 > \sigma\sqrt{\frac{P}{\mathfrak{b}}} \\ \sigma^2\frac{P}{\mathfrak{b}}, & \text{otherwise} \end{array} \right\}. \tag{12}$$

We illustrate the result in Figure 2. Which is analogous to Figure 1, except for the Wigner semi circle is replaced with the Marchenko-Pastur density. We omit the detailed proof of the theorem,



(a) Linear scaling (b) Threshold (c) Square root scaling (d) Threshold

Figure 2: Variation of the spectral norm with batch size. The continuous region corresponds to the fluctuation matrix induced by mini-batching (shown as the Marchenko-Pastur) and the largest eigenvalue of the full Hessian is shown as a single peak. Scaling refers to how the largest eigenvalue decreases as the batch size is increased

but note that it can be proven straightforwardly by combining the multiplicative perturbation results (Benaych-Georges and Nadakuditi, 2011), with the extension of the Marchenko-Pastur law for dependent entries (O'Rourke et al., 2012).

Remark 7 *We note that in the limit of $\sigma^2 \to 1$, ignoring $P, \mathfrak{b}$ by folding them into σ, we have the same results as in (Benaych-Georges and Nadakuditi, 2011). One point to note is that as the noise is multiplicative, a zero fluctuation matrix will of course give all eigenvalues zero.*

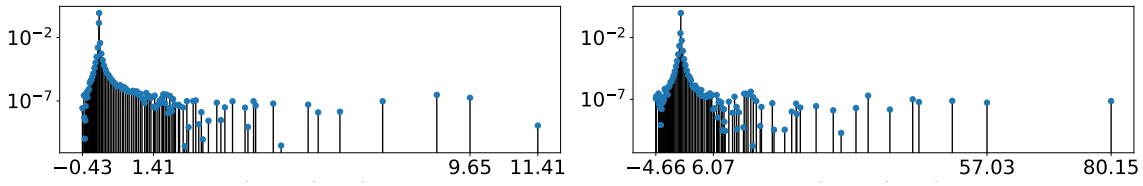
How realistic is the low-rank approximation? Since this is a major assumption in our analysis, we investigate the experimental evidence for the low-rank nature of the empirical Hessian and empirical GGN in Appendix C and provide a theoretical argument for feed forward neural network Hessians.

5. Experimental validation of the theoretical results

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 dataset which does not utilise batch normalisation. We show in Appendix G 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 dataset. To plot the spectrum of the neural network we use Granziol et al. (2019), which gives a discrete moment matched spectral approximation to the underlying spectrum. We use $m = 100$ as the number of moments.

5.1 Effect of spectral broadening for a typical batch size

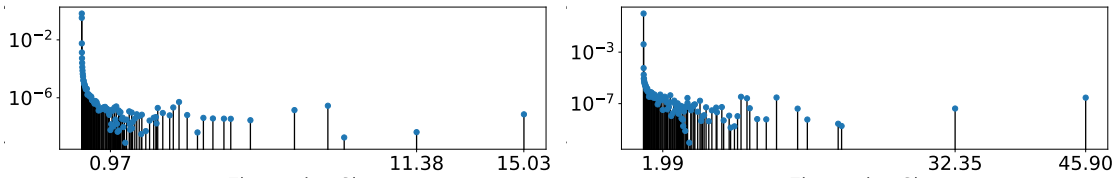
We plot an example effect of the spectral broadening of the Hessian due to mini-batching, for a typical batch size of $B = 128$ in Figure 3. *The magnitude of the extremal eigenvalues are significantly increased as are other outlier eigenvalues, such as the second largest.* We estimate the mean of the continuous region (bulk) of the spectrum as the position where the Ritz[3] weight drops below $1/P$. We see that the spectral width of this continuous region also increases. We plot an example of the



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

Figure 3: Spectral Density of the Hessian at epoch 200, for different sample sizes B, N on a VGG-16 on the CIFAR-100 dataset. The Y-axis corresponds to $p(\lambda)$ and the X-axis to λ.

Generalised Gauss-Newton matrix in Figure 4, which for cross-entropy loss and softmax activation is equal to the Fisher 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 4: 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 dataset. The Y-axis corresponds to $p(\lambda)$ and the X-axis to λ.

5.2 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. We plot the evolution of the Hessian/GGN variance throughout an SGD training cycle in Figure 5. This Figure implies that we expect the batch Hessian extremal eigenvalues to diverge from those of the empirical Hessian during training.

To test this hypothesis we run both SGD and KFAC Martens and Grosse (2015) on the VGG-16 using CIFAR-100 and track the Hessian variance and full Hessian and the average of 10, $B = 128$ batch Hessian extremal eigenvalues and plot the results in Figure 6. The batch Hessian extremal eigenvalues have a large variance. This is to be expected as our results are in the limit $P, B \to \infty$ 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

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

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 σ_ϵ^2 in Theorems 4 & 6

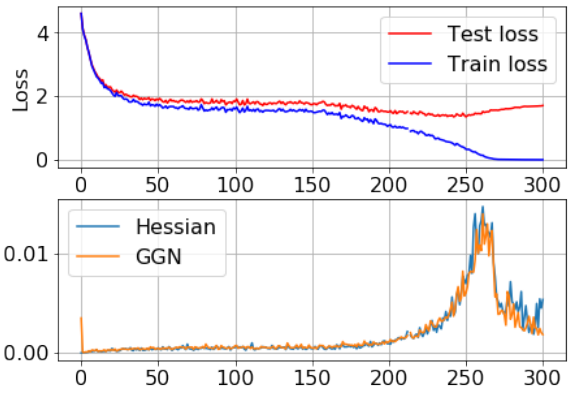
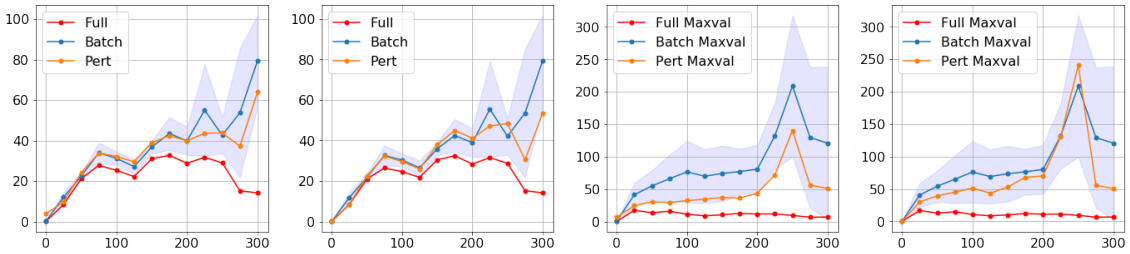


Figure 5: Loss/variance evolution during SGD training for VGG-16 CIFAR-100. Epochs on the X-axis.



(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 6: Evolution of the Variance σ and 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 dataset. Full, Batch and Pert refer to the full, batch and the theoretically predicted Hessian eigenvalues respectively.

multiplicative noise process (Theorem 6) are within 1 standard deviation from the true result. They both follow the increase in variance of the Hessian in Figure 5. We note that the multiplicative noise process gives 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.

6. Applications of the theoretical results

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

$$\delta L(\boldsymbol{w} - \alpha \nabla L) = -\alpha ||\boldsymbol{g}(\boldsymbol{w})||^2 \left(1 - \frac{\alpha \sum_i^P \lambda_i ||\boldsymbol{\phi}_i \boldsymbol{g}(\boldsymbol{w})||^2}{2}\right) \leq -\alpha ||\boldsymbol{g}(\boldsymbol{w})||^2 \left(1 - \frac{\alpha \lambda_1}{2}\right) \tag{13}$$

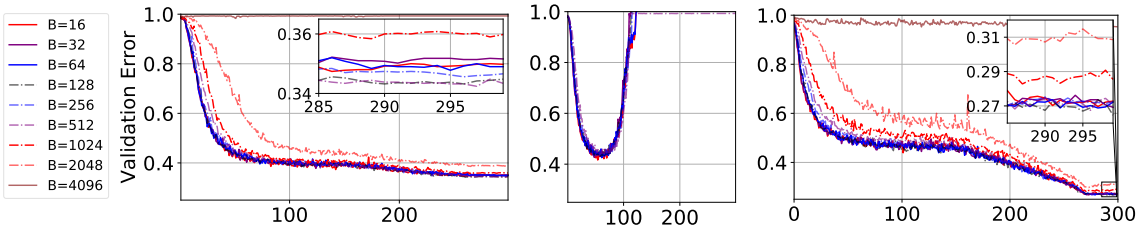
and hence $\alpha < 2/\lambda_1$ must hold to guarantee a decrease in loss. Here $\lambda_1(\boldsymbol{H}_{batch})$, and similarly all outlier eigenvalues of the batch Hessian, are given by Theorem 4. A key term in Equation 13 is the

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[4]), i.e $\lambda_i* \approx \lambda_i + P\sigma^2/B\lambda_i$, is relevant to determining the maximal allowed learning rate. We observe outliers in all our experiments, as shown in Figures 3 and 4, which is consistent with previous literature (Ghorbani et al., 2019; Papyan, 2018).

Large learning rates have been shown to induce implicit regularisation This has for example been 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). Hence learning the largest stable learning rate is an important practical question for neural network training. Given that validation performance is the key metric on which neural network solutions are evaluated we focus on this metric in our experiments.

For small batch sizes the maximal learning rate is proportional to the batch size. As the largest allowable learning rate as shown by Equation 13 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 $B = 128$. We then increase/decrease the batch size by factors of 2, proportionally scaling the learning rate. We plot the results in Figure 7a. The validation



(a) No-BN $\alpha_0 = \frac{0.05B}{128}$, $\gamma = 0$ (b) $\alpha_0 = \frac{0.01B}{128}$ (c) BN $\alpha_0 = \frac{0.1B}{128}$, $\gamma = 5e^{-4}$

Figure 7: Validation error of the VGG-16 architechutre, with and without batch normalisation (BN) on CIFAR-100, with corresponding weight decay γ and initial learning rate α_0.

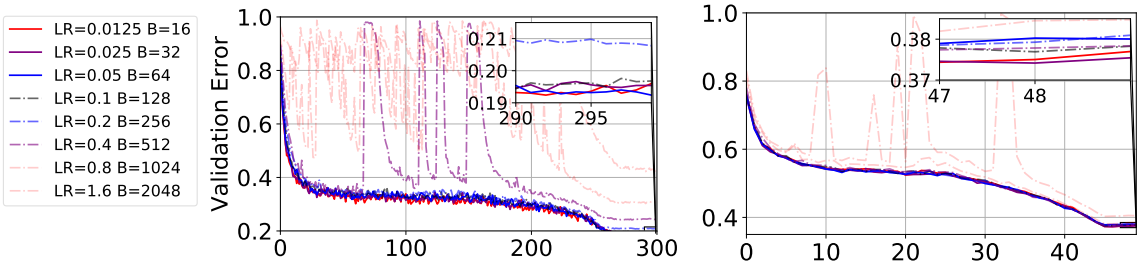
accuracy remains stable for all batch size values, until a small drop for $B = 1024$, a larger drop still for $B = 2048$ and for $B = 4096$ we see no training. As shown in Appendix **??**, the accuracy curves are stable until the limiting batch size of size $B = 1$.

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 5), large learning rates which initially rapidly decrease the loss could become unstable later in training. To see this, we run the same experiment but this time use twice the maximal allowed learning rate. We observe in Figure 7b that initially the loss decreases rapidly (far faster than in the smaller learning rate alternative in Figure 7a), 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.

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

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

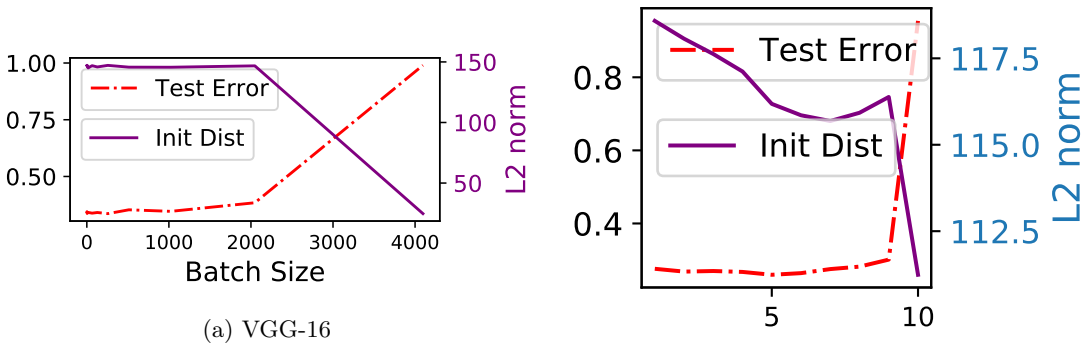
$\gamma = 0.0005$. In this case there is a greater range of permissible learning rates, so we grid search the best learning rate as defined by the validation error for $B = 128$ and use our derived linear scaling rule, as shown in Figure 7c, where we observe a similar pattern. We repeat the experiment on the WideResNet-28 $\times$ 10 (Zagoruyko and Komodakis, 2016) on both the CIFAR-100 and ImageNet 32×32 (Chrabaszcz et al., 2017) datasets shown in Figure 8. Unlike the VGG model without batch normalisation, where unstable trajectories diverge or with batch normalisation do not train. Highly unstable oscillatory WideResNet trajectories converge with learning rate reduction, however they never reach peak performance. The test performance is stable for a variety of learning rates with fixed learning rate to batch size ratio, again confirming the validity of the linear scaling rate rule until a threshold.



(a) CIFAR-100 , $\gamma = 5e^{-4}$

(b) ImageNet-32, $\gamma = 5e^{-5}$

Figure 8: Validation error of the WideResNet-28 $\times$ 10 on the CIFAR-100 and ImageNet32 dataset, with initial learning rate $\alpha_0 = \frac{0.1B}{128}$ and weight decay γ.



(a) VGG-16

(b) VGG-16BN

Figure 9: Test error as a function of initialisation distance for both the VGG-16 and VGG-16BN, for the CIFAR-100 dataset. Learning rate is scaled linearly with batch size.

Alternative learning rate schedules and initialisation distance importance One implicit assumption in Section 6.1 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 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 dataset. 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 3 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 4a 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 4b. We even see the initialisation distance increasing as test error also increases.

6.2 Damping for 2^{nd} order/adaptive optimisation

Damping. Second-order methods minimise the quadratic around a small perturbation of the loss $L(\boldsymbol{w} + \boldsymbol{\delta w})$ i.e

$$\boldsymbol{\delta w}^* = \arg\min_{\boldsymbol{\delta w}} \left(L(\boldsymbol{w}) + \nabla L(\boldsymbol{w})\boldsymbol{\delta w} + \frac{1}{2}\boldsymbol{\delta w}^T \boldsymbol{H}(\boldsymbol{w})\boldsymbol{\delta w} \right) = -\boldsymbol{H}^{-1}(\boldsymbol{w})\nabla L(\boldsymbol{w}). \tag{14}$$

In practice the Hessian is damped, $\boldsymbol{H}^{-1} = \sum_i^P (\lambda_i + \delta)^{-1}\phi_i\phi_i^T$ and δ (the damping coefficient) keeps the optimiser within a trust region (Martens and Sutskever, 2012), limiting its ability to take overly large steps in locally flat directions. It is typically set at constant values and grid searched (Dauphin et al., 2014; Vinyals and Povey, 2012). Hence δ introduces partial adaptivity. At 0 we have a fully Second-order method and at $\delta \gg \lambda_{max}$ we resort to SGD with learning rate α_0/δ. Using our additive noise model from section 3, we can derive an analytic equation for the minimum damping required for a given batch size. The overlap between the batch extremal eigenvectors and their full dataset counterparts is given by $|\boldsymbol{\phi}_i^T\hat{\boldsymbol{\phi}}_i|^2 = 1 - P\sigma^2/\mathfrak{b}\lambda_i^2$ (see Appendix A). By considering the change in loss for a generic Second-order optimiser, writing $\boldsymbol{H}_{emp} = \sum_i \lambda_i \boldsymbol{\psi}_i^T\boldsymbol{\psi}$ and writing the noisy estimated eigenvalue/eigenvector pair from the optimiser as $\lambda_i, \boldsymbol{\phi}_i$, we have

$$L(\boldsymbol{w}_{k+1}) - L(\boldsymbol{w}) = \sum_i^P \frac{\alpha_0 |\boldsymbol{\phi}_i^T \nabla L(\boldsymbol{w})|^2}{\lambda_i + \delta} \left(1 - \frac{\alpha_0}{2(\lambda_i + \delta)} \sum_\mu \lambda_\mu |\boldsymbol{\psi}_u^T \boldsymbol{\phi}_i|^2) \right) \tag{15}$$

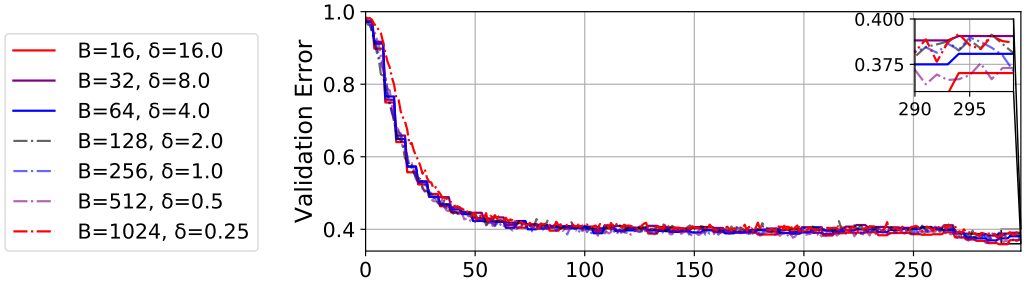
and thus

$$\delta > \frac{\alpha_0 P \sigma^2}{2\mathfrak{b}}. \tag{16}$$

Here we assume δ to be sufficiently large. Hence the largest loss increase is due to estimated flat directions having residual overlap with the sharpest direction of the loss. The total residual overlap is $P\sigma^2/\mathfrak{b}\lambda_i^2$ and for typical positive-definite approximations (Martens, 2016; Vinyals and Povey, 2012; Dauphin et al., 2014), in the worst case, $\lambda_i \ll \delta$ in these directions, from which the result follows. To verify the validity of this relation, we use the second order deep learning optimiser KFAC (Martens and Grosse, 2015) on the VGG-16 with no weight decay, with $\alpha = 1$, grid searching δ until we train to good accuracy. The batch size is increased/decreased by factors of 2 and the damping in inverse proportion. We display the validation errors for KFAC in Figure 10a, where we note good agreement with the theory.

6.3 Square root learning rate scaling for adaptive optimisers with small damping

From Equation 15, for small δ, the greatest loss changes could result from large steps along flat directions. From Theorem 4 we expect the sharpness of some of these directions to grow inversely proportionally to the square root of the batch size. This implies that if our damping is small, we



(a) KFAC Validation

Figure 10: Error of VGG-16 on the CIFAR-100 dataset for the KFAC optimiser for a range of batch sizes and damping coefficients B, δ.

should scale the learning rate as the square root of batch size. We try this for the Adam optimiser with the default damping of 10^{-8}. We grid search the largest stable learning rate for a batch size of 128 and then increase/decrease the batch size by a factor of 2, scaling the learning rate proportional to the square root of the batch size change. We show the results in Figure 11, which validates our hypothesis, whereas the linear prescription fails completely. Using $\alpha = 0.0008, B = 256$ does not result in any training and the results remain as good as random throughout the entire training procedure.

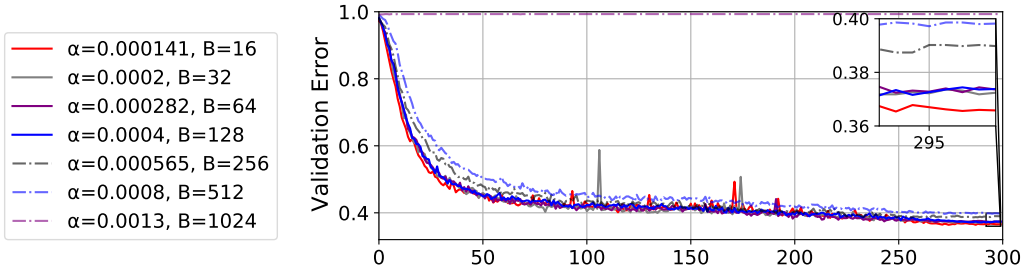


Figure 11: Validation error of Adam with $\delta = 10^{-8}$ (given as ϵ in the original algorithm) on the VGG-16 CIFAR-100 dataset, using the prescribed square root scaling rule of learning rate with batch size.

7. 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) damping in stochastic Second-order methods should be proportional to the ratio of learning rate to batch size. We also note empirically that taking larger steps in the sharp directions of the loss landscape seems to be related to improved generalisation. We extensively validate our predictions and associated implications on the VGG-16 network and CIFAR-100 dataset, 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 datasets. Given that our analysis is neither dataset 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.

References

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.

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.

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.

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.

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

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.

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 eigenvalues and eigenvectors of finite, low rank perturbations of large random matrices. *ICML: Physics in Deep Learning Workshop*, 2018.

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

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.

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.

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.

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.

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

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. *Second-order optimization for neural networks*. PhD thesis, University of Toronto, 2016. URL `http://www.cs.toronto.edu/~jmartens/docs/thesis_phd_martens.pdf`.

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.

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

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.

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.

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.

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

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.

Oriol Vinyals and Daniel Povey. Krylov subspace descent for deep learning. In *Artificial Intelligence and Statistics*, pages 1261–1268, 2012.

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

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

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

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 semi circle 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

$$\boldsymbol{G}_H(z) = (z\boldsymbol{I}_N - \boldsymbol{H})^{-1} \tag{17}$$

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

$$\mathcal{S}_N(z) = \frac{1}{N}\mathrm{Tr}[\boldsymbol{G}_H(z)] \xrightarrow{N\to\infty} \mathcal{S}(z) = \int \frac{\rho(\lambda)}{z-\lambda} du \tag{18}$$

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

$$\mathcal{R}(w) = \mathcal{B}(w) - \frac{1}{w} \tag{19}$$

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 $k \in \mathbb{N}$*

$$\max(E|Z_{i,j}|^k, E|Y_1|^k) < \infty \tag{20}$$

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

$$\begin{cases} \boldsymbol{M}_n(i,i) = Y_i \\ \boldsymbol{M}_n(i,j) = Z_{ij} = \boldsymbol{M}_n(j,i) \end{cases} \tag{21}$$

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 semi circle distribution,*

$$d\mu(\lambda) = \rho(\lambda)d\lambda = \frac{1}{2\pi}\sqrt{4-\lambda^2}\mathbf{1}_{|\lambda|\leq 2}\, d\lambda. \tag{22}$$

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:

$$\mathcal{R}_W(z) = \sigma^2 z. \tag{23}$$

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

$$\psi(H) = \frac{1}{N}\mathrm{Tr}\boldsymbol{H} = \frac{1}{N}\sum_{i=1}^{N}\lambda_i = \int_{\lambda\in\mathcal{D}} d\mu(\lambda)\lambda \tag{24}$$

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 $k \in \mathbb{N}^+$

$$\psi(\boldsymbol{A}^{n_1}\boldsymbol{B}^{n_2}\boldsymbol{A}^{n_3}\boldsymbol{B}^{n_4}) = \psi(\boldsymbol{A}^{n_1})\psi(\boldsymbol{B}^{n_2})\psi(\boldsymbol{A}^{n_3})\psi(\boldsymbol{A}^{n_4}). \tag{25}$$

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 semi circle 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 semi circle eigenvalue distribution can be written as (Tao, 2012)

$$\mathcal{S}_{\boldsymbol{\epsilon}}(z) = \frac{z \pm \sqrt{z^2 - 4\sigma_\epsilon^2}}{2\sigma_\epsilon^2}. \tag{26}$$

From the definition of the Blue transform, we hence have

$$\begin{aligned} z &= \frac{\mathcal{B}_{\boldsymbol{\epsilon}}(z) \pm \sqrt{\mathcal{B}_{\boldsymbol{\epsilon}}^2(z) - 4\sigma_\epsilon^2}}{2\sigma_\epsilon^2} \\ \mathcal{B}_{\boldsymbol{\epsilon}}(z) &= \frac{1}{z} + \sigma_\epsilon^2 z \\ \mathcal{R}_{\boldsymbol{\epsilon}}(z) &= \sigma_\epsilon^2 z. \end{aligned} \tag{27}$$

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:

$$\mathcal{S}_{\boldsymbol{A}}(u) = \frac{1}{N}\frac{1}{u-\lambda_1} + \left(1 - \frac{1}{N}\right)\frac{1}{u} = \frac{1}{u}\left[1 + \frac{1}{N}\frac{\lambda_1}{1-u^{-1}\lambda_1}\right] \tag{28}$$

We can use perturbation theory similar to in Equation (27) to find the Blue and $\mathcal{R}$ transform which to leading order gives

$$\begin{aligned} \mathcal{B}_{\boldsymbol{A}}(\omega) &= \frac{1}{w} + \frac{\lambda_1}{N(1-\omega\lambda_1)} + \mathcal{O}(N^{-2}) \\ \mathcal{R}_{\boldsymbol{A}}(\omega) &= \frac{\lambda_1}{N(1-\omega\lambda_1)} + \mathcal{O}(N^{-2}) \end{aligned} \tag{29}$$

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

$$z = \mathcal{B}_{\boldsymbol{A}}(\mathcal{S}_{\boldsymbol{M}}(z)) + \frac{\lambda_1}{N(1-\lambda_1 \mathcal{S}_{\boldsymbol{M}}(z))} + \mathcal{O}(N^{-2}) \tag{30}$$

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

$$\mathcal{S}_1(z) = -\frac{\lambda_1 \mathcal{S}'_{\epsilon(w)}(z)}{1-\mathcal{S}_{\epsilon(w)}(z)\lambda_1} \tag{31}$$

and hence

$$\mathcal{S}_{\boldsymbol{M}}(z) \approx \mathcal{S}_{\epsilon(w)}(z) - \frac{1}{N}\frac{\lambda_1 \mathcal{S}'_{\epsilon(w)}(z)}{1-\mathcal{S}_{\epsilon(w)}(z)\lambda_1} \tag{32}$$

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

$$\begin{aligned} \mathcal{S}_{\epsilon(w)}(z) &= \frac{1}{\lambda_1} \\ \frac{2\sigma_\epsilon^2}{\lambda_1} &= z \pm \sqrt{z^2 - 4\sigma_\epsilon^2} \\ \therefore z &= \lambda_1 + \frac{\sigma_\epsilon^2}{\lambda_1} \end{aligned} \tag{33}$$

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*

$$|\boldsymbol{\phi}'_i \boldsymbol{\phi}_i|^2 = \left\{ \begin{array}{ll} 1 - \frac{P}{\mathfrak{b}}\frac{\sigma_\epsilon^2}{\lambda_1^2}, & \text{if } \lambda_1 > \sqrt{\frac{P}{\mathfrak{b}}}\sigma_\epsilon \\ 0, & \text{otherwise} \end{array} \right\} \tag{34}$$

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 semi circle law $p(\lambda) = \frac{\sqrt{4\sigma_\epsilon^2 - \lambda^2}}{2\pi\sigma_\epsilon^2}$. Then we have*

$$|\boldsymbol{\phi}'_i \boldsymbol{\phi}_i|^2 = \left\{ \begin{array}{ll} 1 - \frac{\sigma_\epsilon^2}{\lambda_1^2}, & \text{if } \lambda_1 > 2\sigma_\epsilon \\ 0, & \text{otherwise} \end{array} \right\} \tag{35}$$

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. Note that Feier (2012) use a different convention for the Stieltjes transform

$$\mathcal{S}_P(z) = \int_{\mathbb{R}} \frac{1}{x-z}\rho(x)dx = \frac{1}{P}\operatorname{Tr}(\boldsymbol{M}_n/\sqrt{P} - z\boldsymbol{I})^{-1} \tag{36}$$

We consider a series of matrices

$$\boldsymbol{X}_N = \left(r_i^s / \sqrt{P} \right)_{1 \leq i \leq P,\, 1 \leq s \leq N} \tag{37}$$

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 $\boldsymbol{W}_P^s$

$$\frac{1}{P} \operatorname{Tr}(\boldsymbol{W}_P^s - z\boldsymbol{I})^{-1} = -\frac{1}{P} \sum_{k=0}^{\infty} \frac{\operatorname{Tr}(\boldsymbol{W}_P^s)^k}{z^{k+1}} = -\frac{1}{P} \left(\frac{P-1}{z} + \frac{1}{z - \beta\sigma^2} \right) \tag{38}$$

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 $|z| \sim -\frac{1}{z}$

$$\begin{aligned} z &= \frac{P(s\beta\sigma^2 - 1) \pm \sqrt{P^2(s\beta\sigma^2 - 1)^2 + 4Ps(P-1)\beta\sigma^2}}{2Ps} = \frac{P(s\beta\sigma^2 - 1) \pm \sqrt{P^2(s\beta\sigma^2 + 1)^2 - 4Ps\beta\sigma^2}}{2Ps} \\ z &\approx \frac{P(s\beta\sigma^2 - 1) - P(s\beta\sigma^2 + 1) - \frac{2Ps\beta\sigma^2}{s\beta\sigma^2 + 1}}{2Ps} = -\frac{1}{s} + \frac{\beta\sigma^2}{P(s\beta\sigma^2 + 1)} \end{aligned} \tag{39}$$

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:

$$\begin{aligned} \mathcal{R}_{\boldsymbol{W}_P}(s) &= N \times \left(z - \frac{1}{s} \right) = \frac{N\beta\sigma^2}{P(s\beta\sigma^2 + 1)} = \frac{\sigma^2}{(s\beta\sigma^2 + 1)} \\ \mathcal{B}_{\boldsymbol{W}_P}(s) &= z = -\frac{1}{s} + \frac{\sigma^2}{(s\beta\sigma^2 + 1)} \end{aligned} \tag{40}$$

and hence

$$\mathcal{S}_{\boldsymbol{W}_P} = \frac{-(z + \sigma^2(1-\beta)) + \sqrt{(z + \sigma^2(1-\beta))^2 - 4\beta\sigma^2 z}}{2\beta\sigma^2 z} \tag{41}$$

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

$$\rho(y) = \frac{\sqrt{4\beta\sigma^2 y - (y + \sigma^2(1-\beta))^2}}{2\beta\sigma^2 y} \tag{42}$$

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).

$$\mathcal{T}(z) = \frac{z - \sigma^2(1+\beta) - \sqrt{(z + \sigma^2(1-\beta))^2 - 4\beta\sigma^2 z}}{2\beta\sigma^2}. \tag{43}$$

Appendix C. 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 datasets. However theoretical arguments, rely on the Generalised Gauss-Newton matrix decomposition. From equation 10 it can be surmised that the rank of the GGN is upper bounded by $N \times d_y$ (the dataset 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 dataset size becomes large, such as ImageNet with 10^7 and class number also large, even the GGN bound is ineffective. We hence provide what is to our knowledge a novel theoretical argument for a Hessian rank bound for feed forward neural networks with cross-entropy loss in Section C.4. The key intuition for the proof is that each product of weights is a rank one object. Hence if the sum of these products can be bounded we can bound the rank. Since the sum depends on the number of neurons, the rank bound can end up becoming very small.

C.1 Experimental Validation of Low Rank Approximation

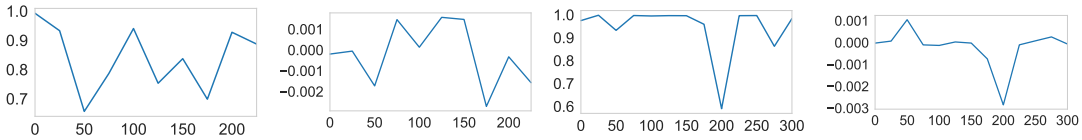
A full Hessian inversion with computational cost $\mathcal{O}(P^3)$ is infeasible for large neural networks. Hence, counting the number of 0 eigenvalues (which sets the degeneracy) is not feasible in this manner. Furthermore, there would still be issues with numerical precision, so a threshold would be needed for accurate counting. Hence, based on our understanding of the Lanczos algorithm, discussed in Appendix F, 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 $\boldsymbol{H}$. Each eigenvalue, eigenvector pair estimated by the Lanczos algorithm is called a Ritz-value/Ritz-vector. We hence take $m \gg 1$, where typically and for consistency we take $m = 100$ in our experiments. We then take the Ritz value closest to the origin and take that as a proxy for the 0 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, is that the spectral mass near the origin, may split into multiple components and 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 dataset shown in Figure 12. 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.

C.2 VGG16

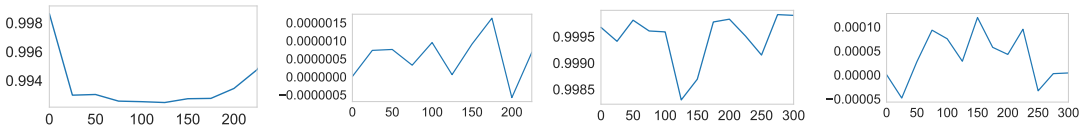
For the VGG-16, which forms the reference model for this paper, we see that for both the Generalised Gauss-Newton matrix (shown in Figure 13a) and the Hessian (shown in Figure 13c) that the rank



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

Figure 12: Rank degeneracy $\mathcal{D}$ (proportion of zero eigenvalues) evolution throughout training using the VGG-16 and PreResNet-110 on the CIFAR-100 dataset, the weight corresponds to the spectral mass of the Ritz value(s) considered to correspond to $\mathcal{D}$

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 for GPU precision, as shown in Figure 13b. For the Hessian, for which we combine the two smallest absolute value Ritz values, we have as expected an even larger spectral degeneracy. The weighted average, also gives a value very close to 0, as shown in Figure 13d. Although the combined weighted average is much closer to the origin, than that of the lone spectral peak, shown in Figure 12, 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 be dominated completely by numerical imprecision.

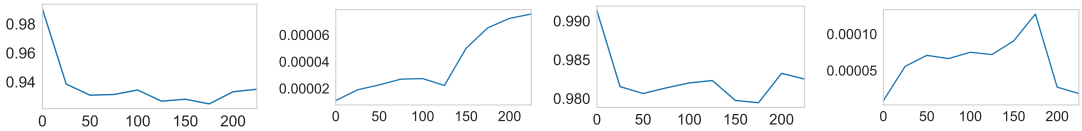


(a) GGN Degeneracy (b) GGN Ritz Value (c) Hessian Degeneracy (d) Hessian Ritz Value

Figure 13: Rank degeneracy (proportion of zero eigenvalues) evolution throughout training using the VGG-16 on the CIFAR-100 dataset, total training 225 epochs, the Ritz value corresponds to the value of the node which we assign to 0

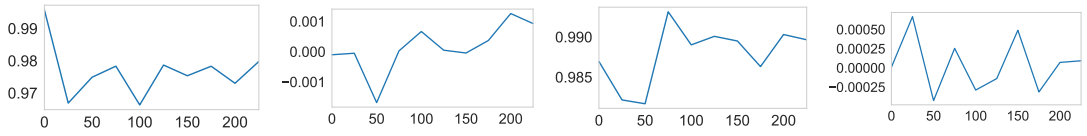
C.3 PreResNet110

We repeat the same experiments in section C.2 for the preactivate residual network with 110 layers, on the same dataset. The slight subtlety is that as explained in Section G, 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 Degeneracy (b) BN-train Ritz Value (c) BN-eval Degeneracy (d) BN-eval Ritz Value

Figure 14: Generalised Gauss-Newton matrix rank degeneracy (proportion of zero eigenvalues) evolution throughout training using the PreResNet-110 on the CIFAR-100 dataset, total training 225 epochs, the Ritz value corresponds to the value of the node which we assign to 0



(a) BN-train Degeneracy (b) BN-train Ritz Value (c) BN-eval Degeneracy (d) BN-eval Ritz Value

Figure 15: Hessian rank degeneracy (proportion of zero eigenvalues) evolution throughout training using the PreResNet-110 on the CIFAR-100 dataset, total training 225 epochs, the Ritz value corresponds to the value of the node which we assign to 0

C.4 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

$$\boldsymbol{z}(\boldsymbol{x}_i; \boldsymbol{w})_q = f(\boldsymbol{W}_H^T f(\boldsymbol{W}_{H-1}^T ....f(\boldsymbol{W}_1 \boldsymbol{x}))) = \prod_{l=0}^{H} \sum_{n_{i,l}=1}^{N_l} \sum_{i}^{d_x} \boldsymbol{x}_i \boldsymbol{w}_{n_{i,l}, n_{i,l+1}} \tag{44}$$

$\boldsymbol{w}_{n_{i,l}, n_{i,l+1}}$ 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 of the loss in the small loss limit tends to

$$\frac{\partial^2 \ell(h(\boldsymbol{x}_i; \boldsymbol{w}), \boldsymbol{y}_i)}{\partial w_{\phi,\kappa} \partial w_{\theta,\nu}} \to - \sum_{m \neq c} \exp(h_m) \left[\frac{\partial^2 h_m}{\partial w_{\phi,\kappa} \partial w_{\theta,\nu}} + \frac{\partial h_m}{\partial w_{\phi,\kappa}} \frac{\partial h_m}{\partial w_{\theta,\nu}} \right] \tag{45}$$

$$\begin{aligned} \left[\frac{\partial^2 h_m}{\partial w_{\phi,\kappa} \partial w_{\theta,\nu}} + \frac{\partial h_m}{\partial w_{\phi,\kappa}} \frac{\partial h_m}{\partial w_{\theta,\nu}} \right] &= \prod_{l=1}^{d-1} \sum_{n_{i,l} \neq [(\phi,\kappa),(\theta,\nu)]}^{N_{i,l}} \sum_{i}^{d_x} \boldsymbol{x}_i \boldsymbol{w}_{n_{i,l}, n_{i,l+1}} \\ &+ \left(\prod_{l=1}^{d-1} \sum_{n_{i,l} \neq (\theta,\nu)}^{N_{i,l}} \sum_{i}^{d_x} \boldsymbol{x}_i \boldsymbol{w}_{n_{i,l}, n_{i,l+1}} \right) \left(\prod_{l=1}^{d-1} \sum_{n_{j,l} \neq (\phi,\kappa)}^{N_{j,l}} \sum_{i}^{d_x} \boldsymbol{x}_i \boldsymbol{w}_{n_{j,l}, n_{j,l+1}} \right) \end{aligned} \tag{46}$$

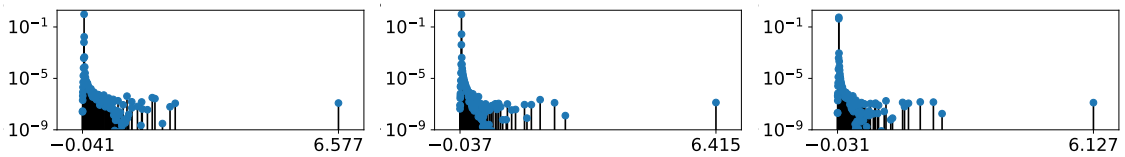
Each product of weights contributes an object of rank-1 (as shown in section 2). Furthermore, the rank of a product is the minimum of the constituent ranks, i.e $\text{rank}(AB) = \min \text{rank}(A, B)$. Hence Equation 46 is rank bounded by a $2(\sum_l N_l + d_x)$, where N_l is the total numbers of neurons in the network. By rewriting the loss per-sample and repeating the same arguments and including the class factor

$$\frac{\partial^2 \ell}{\partial w_k \partial w_l} = - \frac{\partial^2 h_{q(i)}}{\partial w_k \partial w_l} + \frac{\sum_j \exp(h_j) \sum_i \exp(h_i) (\frac{\partial^2 h_i}{\partial w_k \partial w_l} + \frac{\partial h_i}{\partial w_k} \frac{\partial h_i}{\partial w_l}) - \sum_i \exp(h_i) \frac{\partial h_i}{\partial w_k} \sum_j \frac{\partial h_j}{\partial w_l} \exp(h_j)}{[\sum_j \exp(h_j)]^2} \tag{47}$$

We obtain 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 dataset, for the CIFAR-10 dataset, 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 $1 - \frac{577{,}600}{1.6 \times 10^7} = 0.9639$. We give extensive experimental validation for the low-rank nature of the empirical Hessian in Section C.

Appendix D. True Loss Surface

As a small aside, in our work we consider the empirical Hessian and the batch Hessian. Another natural object which emerges from our analysis, which was partially investigated in Granziol et al. (2018) is the Hessian under the data generating distribution. For P finite 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. Similarly in this limit our empirical risk converges almost surely to our true risk, i.e. we eliminate the *generalization gap*. However, in deep learning typically the network size eclipses the dataset size by orders of magnitude.[5] This is similar to considering the 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).



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

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

As the true Hessian and true risk are unobservable, we consider whether the empirical Hessian provides a valid approximation to the true Hessian, by evaluating the Hessian by artificially increasing the number of samples through the use of data-augmentation, explained in section 5. As shown in Figure 16, the extremal eigenvalues are reduced in size as the sample number is increased, in accordance with Theorem 4.

In order to simulate the effect of increasing the data-set size, we use data augmentation to artificially increase the dataset size, specifically random horizontal flips, 4×4 zero padding and random 32×32 crops and then we use the Pearlmutter trick (Pearlmutter, 1994) on the augmented dataset 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 dataset to be equivalent to an independent set of larger size than the original dataset. This intuition is grounded as training without augmentation leads to significant performance decreases similar to reducing the dataset size. Specifically for the VGG-16 without augmentation we achieve a testing accuracy of 48.8% compared to 72.1% on the CIFAR-100 dataset running the same schedule. We note from Figure 16, 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 remains unaffected. This gives evidence that the true Hessian may be similar to the empirical Hessian. This is a different conclusion to that reached in (Granziol et al., 2018).

Appendix E. Experimental Details

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 dataset (45,000 training samples and 5,000 validation

5. CIFAR datasets, which have $50,000$ examples, are routinely used to train networks with about 50 million parameters.

samples) using SGD and K-FAC optimisers. For both SGD and K-FAC, we use the following learning rate schedule:

$$\alpha_t = \begin{cases} \alpha_0, & \text{if } \frac{t}{T} \leq 0.5 \\ \alpha_0[1 - \frac{(1-r)(\frac{t}{T}-0.5)}{0.4}] & \text{if } 0.5 < \frac{t}{T} \leq 0.9 \\ \alpha_0 r, & \text{otherwise} \end{cases} \tag{48}$$

We use learning rate ratio $r = 0.01$ and total number of epochs budgeted $T = 300$. We further use momentum $\rho = 0.9$, a weight decay coefficient of 0.0005 and data-augmentation on PyTorch (Paszke et al., 2017). We further set the inversion frequency to be once per 100 iterations for K-FAC.

Appendix F. 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 dataset 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,

$$\int_a^b f(\lambda) d\mu(\lambda) = \sum_{j=1}^{M} \rho_j f(t_j) + R[f] \tag{49}$$

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 tri-diagonal 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*

$$\begin{aligned} \lambda_1 \geq \theta_1 \geq \lambda_1 - \frac{(\lambda_1 - \lambda_n)\tan^2(\theta_1)}{(c_{k-1}(1+2\rho_1))^2} \\ \lambda_n \leq \theta_k \leq \lambda_m + \frac{(\lambda_1 - \lambda_n)\tan^2(\theta_1)}{(c_{k-1}(1+2\rho_1))^2} \end{aligned} \tag{50}$$

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*

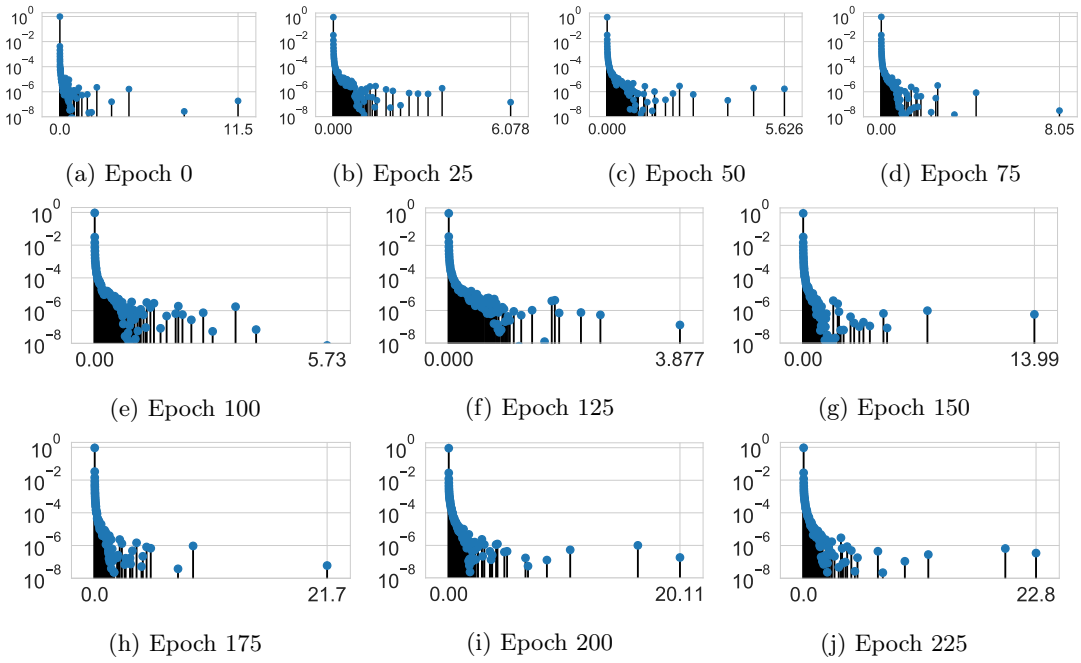
Proof: See (Golub and Meurant, 1994). The first term on the RHS of Equation 49 using Theorem 13 can be seen as a discrete approximation to the spectral density matching the first m moments $v^T H^m v$ (Golub and Meurant, 1994; Golub and Van Loan, 2012), where v is the initial seed vector. Using the expectation of quadratic forms, for zero mean, unit variance random vectors, using the linearity of trace and expectation

$$\mathbb{E}_v \mathrm{Tr}(v^T H^m v) = \mathrm{Tr}\mathbb{E}_v(vv^T H^m) = \mathrm{Tr}(H^m) = \sum_{i=1}^{N} \lambda_i = N \int_{\lambda \in \mathcal{D}} \lambda d\mu(\lambda) \tag{51}$$

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 G. 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

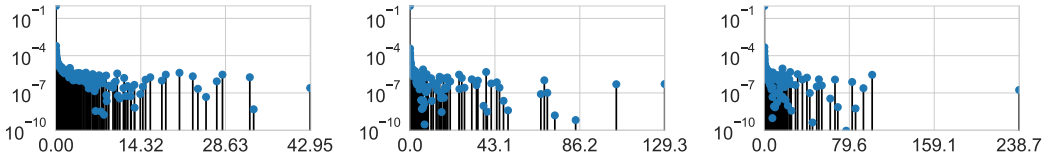
(h) Epoch 175 (i) Epoch 200 (j) Epoch 225

Figure 17: Generalised Gauss-Newton matrix full empirical spectrum for the PreResNet-110 on the CIFAR-100 dataset, 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.

G.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 17 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 18, for comparison. Where we see significant broadening.

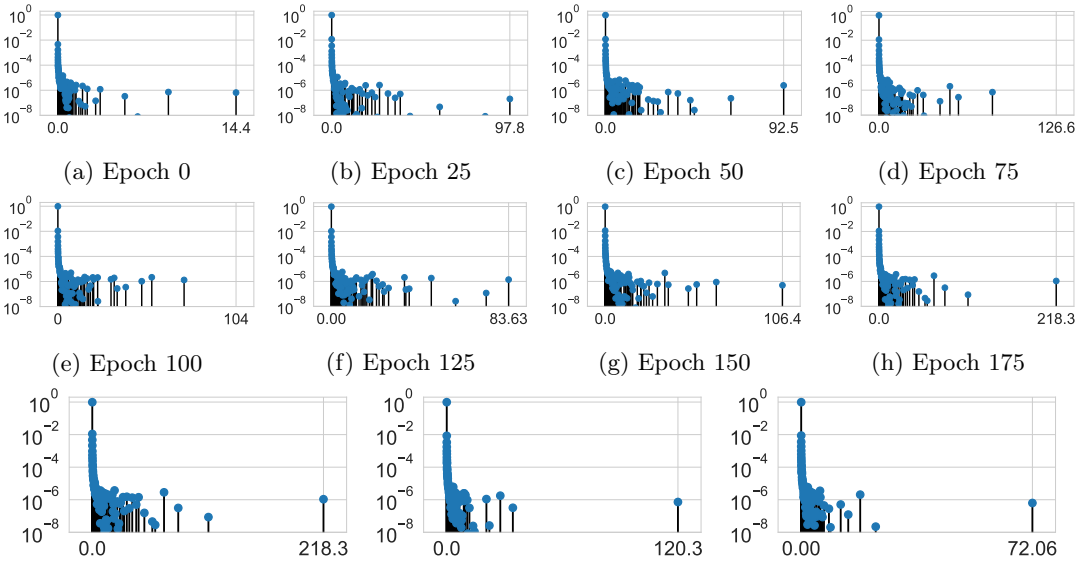


(a) Epoch 175, $B = 128$ (b) Epoch 200, $B = 128$ (c) Epoch 225, $B = 128$

Figure 18: Generalised Gauss-Newton matrix full empirical spectrum for the PreResNet-110 on the CIFAR-100 dataset, total training 225 epochs, batch norm train mode, samples taken with a batch of $B = 128$

G.2 Generalised Gauss-Newton matrix - Evaluation Mode

Similarly to the previous section, we show in Figure 20 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 dataset, shown in Figure **??**. Again although the results are stochastic, with large variance the trend is consistent.

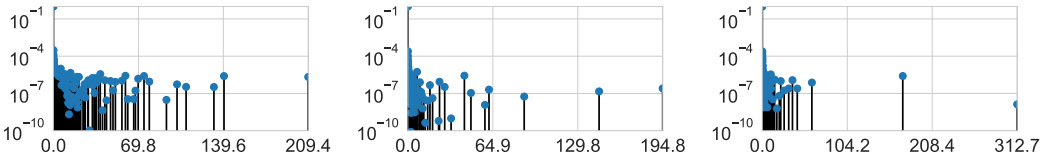


(a) Epoch 0 (b) Epoch 25 (c) Epoch 50 (d) Epoch 75

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

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

Figure 19: Generalised Gauss-Newton matrix full empirical spectrum for the PreResNet-110 on the CIFAR-100 dataset, total training 225 epochs, batch norm eval mode



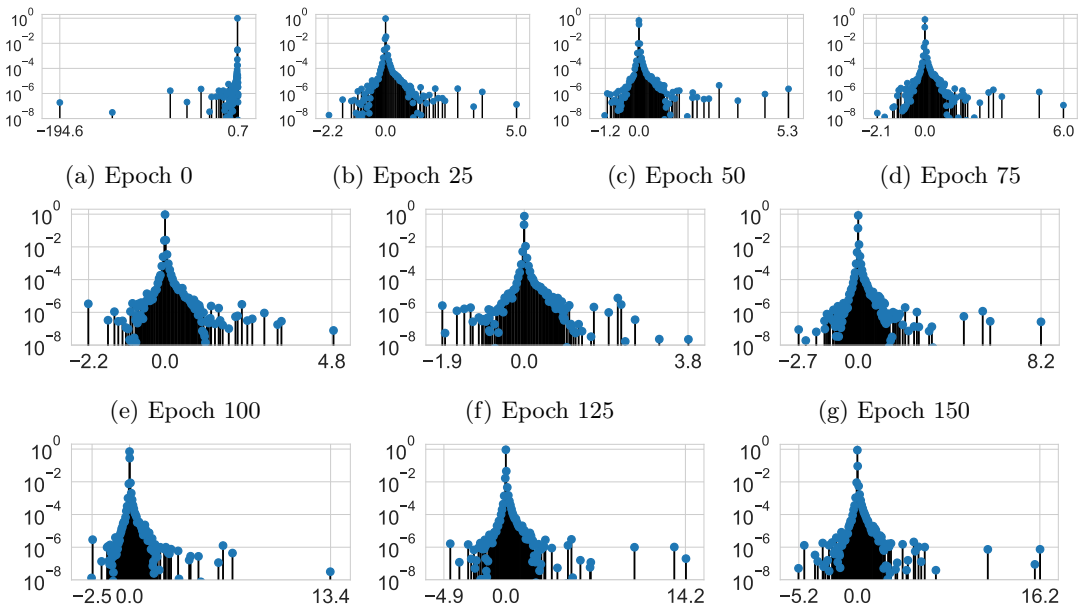
(a) Epoch 175, $B = 128$ (b) Epoch 200, $B = 128$ (c) Epoch 225, $B = 128$

Figure 20: Generalised Gauss-Newton matrix full empirical spectrum for the PreResNet-110 on the CIFAR-100 dataset, total training 225 epochs, batch norm eval mode, $B = 128$ sub-sampled spectrum

G.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.

Similarly at all points in training, stochastic batch Hessians are shown to be significantly broadened,



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

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

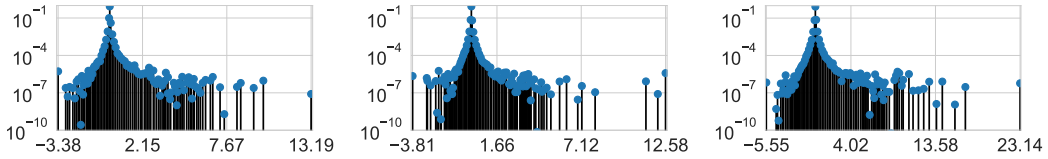
(h) Epoch 175 (i) Epoch 200 (j) Epoch 225

Figure 21: Hessian full empirical spectrum for the PreResNet-110 on the CIFAR-100 dataset, total training 225 epochs, batch norm train mode

we see this by comparing the full data empirical Hessian spectrum 21 compared to the Hessian at the same point in weight space but using only a batch size of $B = 128$ in Figure 22.

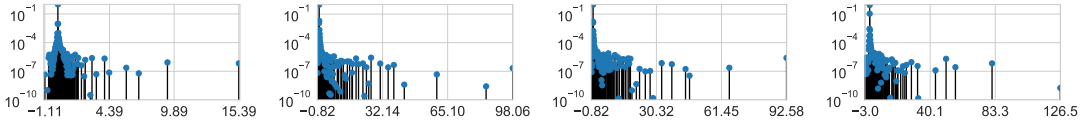
G.4 Hessian - Batch Normalisation Evaluation Mode

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 24 compared to the Hessian at the same point in weight space but using only a batch size of $B = 128$ in Figure 25.



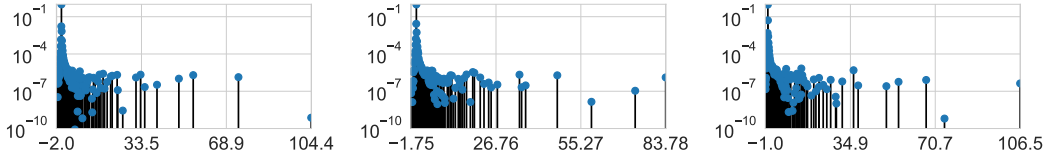
(a) Epoch 100, $B = 128$ (b) Epoch 125, $B = 128$ (c) Epoch 150, $B = 128$

Figure 22: Hessian batch spectrum for the PreResNet-110 on the CIFAR-100 dataset, total training 225 epochs, batch norm train mode, $B = 128$



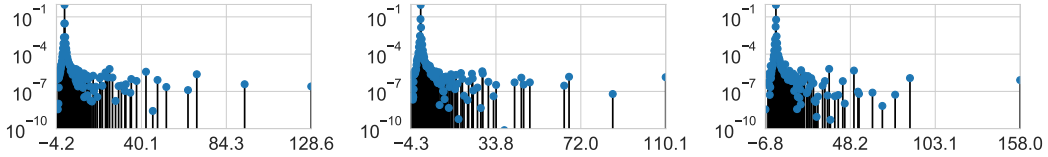
(a) Epoch 0 (b) Epoch 25 (c) Epoch 50 (d) Epoch 75

Figure 23: Hessian full empirical spectrum for the PreResNet-110 on the CIFAR-100 dataset, total training 225 epochs, batch norm evaluation mode



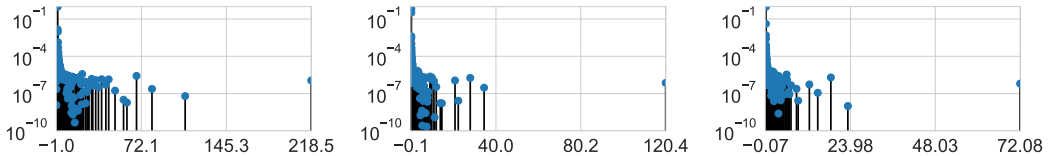
(a) Epoch 100 (b) Epoch 125 (c) Epoch 150

Figure 24: Hessian full empirical spectrum for the PreResNet-110 on the CIFAR-100 dataset, total training 225 epochs, batch norm evaluation mode



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

Figure 25: Hessian batch spectrum for the PreResNet-110 on the CIFAR-100 dataset, total training 225 epochs, batch norm train mode, $B = 128$



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

Figure 26: Hessian full empirical spectrum for the PreResNet-110 on the CIFAR-100 dataset, total training 225 epochs, batch norm evaluation mode