

A PROOF OF MAIN THEOREM

Proof. The cross-entropy loss $\ell(h(\mathbf{x}_i; \mathbf{w}), \mathbf{y})$ of a single sample \mathbf{x}_i is defined by:

$$\ell(h(\mathbf{x}_i; \mathbf{w}), \mathbf{y}_i) = - \sum_c^{d_y} (1 - \mathbb{1}[h(\mathbf{x}_i; \mathbf{w})_c \neq y_c]) \log h(\mathbf{x}_i; \mathbf{w})_c \quad (14)$$

Where d_y is the number of classes and $\mathbb{1}$ is the indicator function which takes the value of 1 for the incorrect class and 0 for the correct class, $\mathbf{z}(\mathbf{x}_i; \mathbf{w})$ is the softmax input. The softmax output $\mathbf{z}(\mathbf{x}_i; \mathbf{w})_c$ for class c is given by

$$h(\mathbf{x}_i, \mathbf{w})_c = \sigma(\mathbf{z})_c = \frac{\exp \mathbf{z}(\mathbf{x}_i; \mathbf{w})_c}{\sum_{k=1}^{d_y} \exp \mathbf{z}(\mathbf{x}_i; \mathbf{w})_k} = \left(1 + \sum_{k \neq c}^{d_y} \exp(\mathbf{z}(\mathbf{x}_i; \mathbf{w})_k - \mathbf{z}(\mathbf{x}_i; \mathbf{w})_c) \right)^{-1} \quad (15)$$

Hence combining Equations 14 and 15, the loss per sample can be written as

$$\ell(h(\mathbf{x}_i; \mathbf{w}), \mathbf{y}_i) = \log \left(1 + \sum_{k \neq c(i)}^{d_y} \exp(z_{k,c(i)}) \right) \quad (16)$$

in which $c(i)$ denotes the correct class for the data point \mathbf{x}_i and $z_{k,c(i)} = \mathbf{z}(\mathbf{x}_i; \mathbf{w})_k - \mathbf{z}(\mathbf{x}_i; \mathbf{w})_{c(i)}$. Note that, for the per sample loss $\ell(h(\mathbf{x}_i; \mathbf{w}), \mathbf{y}_i) \rightarrow 0$, $\exp(z_{k,c(i)}) \rightarrow 0 \forall k \neq c(i)$. Using the chain rule

$$\begin{aligned} \frac{\partial^2 \ell(h(\mathbf{x}_i; \mathbf{w}), \mathbf{y}_i)}{\partial w_l \partial w_m} &= \frac{\sum_{k \neq c(i)} \exp(z_{k,c(i)}) \left[\frac{\partial^2 z_{k,c(i)}}{\partial w_l \partial w_m} + \frac{\partial z_{k,c(i)}}{\partial w_l} \frac{\partial z_{k,c(i)}}{\partial w_m} \right]}{1 + \sum_{i \neq c} \exp(z_{k,c(i)})} \\ &\quad - \frac{\sum_{k \neq c(i), u \neq c(i)} \exp(z_{k,c(i)}) \frac{\partial z_{k,c(i)}}{\partial w_l} \exp(z_{u,c(i)}) \frac{\partial z_{u,c(i)}}{\partial w_m}}{(1 + \sum_{k \neq c(i)} \exp(z_{k,c(i)}))^2} \end{aligned} \quad (17)$$

As shown in Milne (2019), the network is differentiable at the majority of points in weight space. Specifically the zero set of non-zero real analytic functions (the network is piecewise analytic in the weights) has Lebesgue measure zero. Hence all we need to show is that the output derivatives tend to ∞ more slowly than the loss tends to 0. To do this we consider

$$\frac{\partial^m}{\prod_m \partial w_{\mu_m, \phi_m}^m} \sum_{i=1}^{d_x} \sum_{j=1}^{\gamma} \mathbf{x}_i A_{i,j} \prod_{k=1}^H w_{i,j}^{(k)} = \sum_{i=1}^{d_x} \mathbf{x}_i A_{i,j} \sum_{k \neq m}^{\gamma / \prod_m n_m} w_{i,j}^{(k)} (1 - \mathbb{1}[w_{\mu_m, \phi_m}^m]) \quad (18)$$

We are interested in the limit where the output for the correct class $\mathbf{z}(\mathbf{x}_i; \mathbf{w})_{c(i)} \rightarrow \infty$. In which case, at least one of the $w_{i,j}^{(k)}$ must diverge. The dependence on the weights $w_{i,j}^{(k)}$ of $\mathbf{z}(\mathbf{x}_i, \mathbf{w})$ and its derivatives (see (18)) is clearly polynomial. It follows immediately that, in the studied zero-loss limit, $\exp(\mathbf{z}(\mathbf{x}_i, \mathbf{w})_q) \rightarrow 0$ exponentially quickly in the diverging $w_{i,j}^{(k)}$, and so

$$\frac{\partial^m \mathbf{z}(\mathbf{x}_i; \mathbf{w})_q}{\prod_m \partial w_{\mu_m, \phi_m}^m} \exp(\mathbf{z}(\mathbf{x}_i; \mathbf{w})_q) \rightarrow 0 \quad (19)$$

also exponentially quickly (though with some unimportant polynomial pre-factor).

Hence in the limit $\ell(h(\mathbf{x}_i; \mathbf{w}), \mathbf{y}_i) \rightarrow 0$, all terms in Equation 17 have norms tending to zero. Hence $\frac{\partial^2 \ell(h(\mathbf{x}_i; \mathbf{w}), \mathbf{y}_i)}{\partial w_l \partial w_m} \rightarrow 0$. Taking $l = m$ and summing over m we have 0 trace and using the Frobenius norm identity, i.e. taking the sum of squares over l, m we have $\sum_i^P \lambda_i^2 \rightarrow 0$ and hence $\lambda_1 \rightarrow 0$.

Remark. By writing the loss in terms of the activation σ at the output of the final layer $f(\mathbf{w})$, i.e $L(\mathbf{w}) = \sigma(f(\mathbf{w}))$. The Hessian may be expressed using the chain rule as

$$\mathbf{H}(\mathbf{w})_{jk} = \frac{1}{N} \sum_{n=1}^N \left(\sum_{c=0}^{d_y} \sum_{l=0}^{d_y} \frac{\partial^2 \sigma(f(\mathbf{w}))}{\partial f_l(\mathbf{w}) \partial f_c(\mathbf{w})} \frac{\partial f_l(\mathbf{w})}{\partial w_j} \frac{\partial f_c(\mathbf{w})}{\partial w_k} + \sum_{c=0}^{d_y} \frac{\partial \sigma(f(\mathbf{w}))}{\partial w_j} \frac{\partial^2 f_c(\mathbf{w})}{\partial w_j \partial w_k} \right) \quad (20)$$

Where, for the cross-entropy loss and softmax output at exactly 0 loss, $\frac{\partial f_l(\mathbf{w})}{\partial w_j} = 0$ and $\frac{\partial^2 \sigma(f(\mathbf{w}))}{\partial f_l(\mathbf{w}) \partial f_c(\mathbf{w})} = 0$. However, in practice, since the weights are finite, we never have 0 loss. Hence, unlike our proof which shows that the Hessian is given by a product of a polynomial and exponential in the weights which we expect to go to 0 in the limit of large weights and low loss, this simple result does yield information prior to the loss being exactly 0. \square

B SPECTRAL PLOTS

B.1 MLP

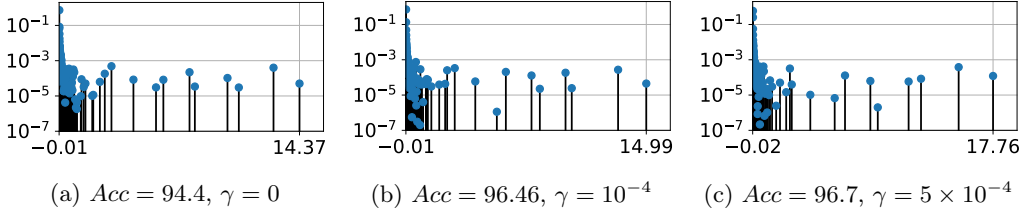


Figure 3: Hessian spectrum for MLP after 50 epochs of SGD on the MNIST dataset, for various $L2$ regularisation coefficients γ

B.2 CNN

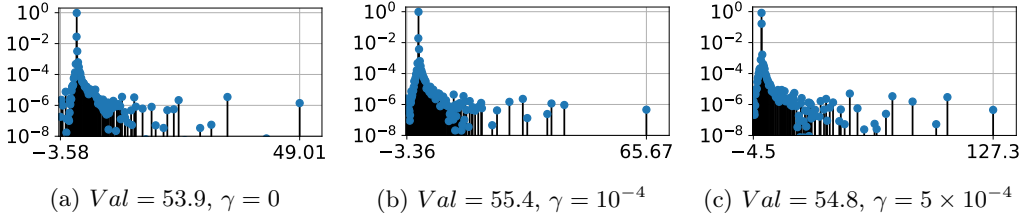


Figure 4: Hessian spectrum for CNN after 300 epochs of SGD on the CIFAR-100 dataset, for various $L2$ regularisation coefficients γ

B.3 PRERESNET164

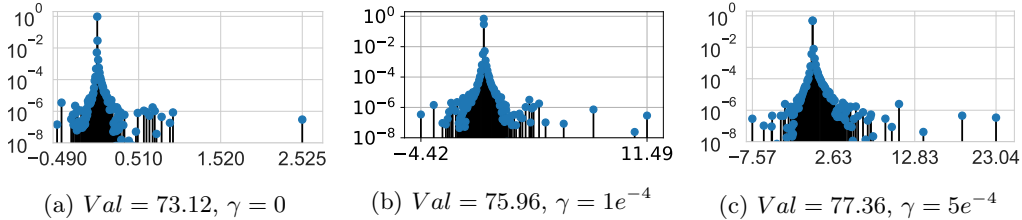


Figure 5: Hessian spectrum for PreResNet-164 after 300 epochs of SGD on the CIFAR-100 dataset, for various $L2$ regularisation coefficients γ

B.4 WIDERESNET

C EXPERIMENT DETAILS

C.1 IMAGE CLASSIFICATION EXPERIMENTS

Hyper parameter Tuning For SGD and Gadam, we set the momentum parameter to be 0.9 whereas for Adam, we set $(\beta_1, \beta_2) = (0.9, 0.999)$ and $\epsilon = 10^{-8}$, their default values.

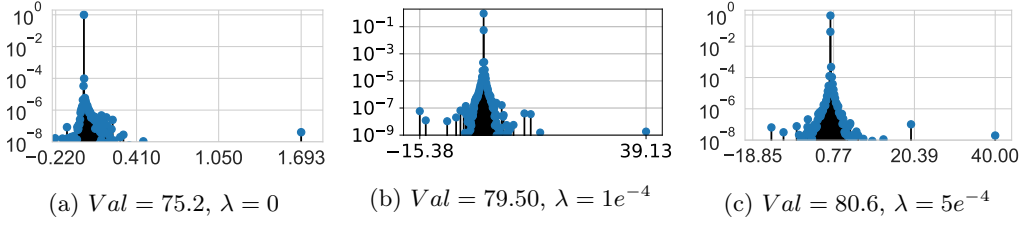


Figure 6: Hessian spectrum for WideResNet28 \times 10 after 300 epochs of SGD on the CIFAR-100 dataset, for various $L2$ regularisation coefficients γ , Batch Norm Train mode

For SGD, we use a grid searched initial learning rates in the range of $[0.01, 0.03, 0.1]$ for all experiments with a fixed weight decay; for Adam and all its variants, we use grid searched initial learning rate range of $[10^{-4}, 3 \times 10^{-3}, 10^{-3}]$. After the best learning rate has been identified, we conduct a further search on the weight decay, which we find often leads to a trade off between the convergence speed and final performance. For CIFAR experiments, we search in the range of $[10^{-4}, 10^{-3}]$ whereas for ImageNet experiments, we search in the range of $[10^{-6}, 10^{-5}]$. For decoupled weight decay, we search the same range for the weight decay scaled by initial learning rate.

C.2 EXPERIMENTAL DETAILS

For all experiments with SGD, we use the following learning rate schedule for the learning rate at the t -th epoch, similar to Izmailov et al. (2018):

$$\alpha_t = \begin{cases} \alpha_0, & \text{if } \frac{t}{T} \leq 0.5 \\ \alpha_0[1 - \frac{(1-r)(\frac{t}{T}-0.5)}{0.4}] & \text{if } 0.5 < \frac{t}{T} \leq 0.9 \\ \alpha_0 r, & \text{otherwise} \end{cases} \quad (21)$$

where α_0 is the initial learning rate. In the motivating logistic regression experiments on MNIST, we used $T = 50$. $T = 300$ is the total number of epochs budgeted for all CIFAR experiments. We set $r = 0p01$ for all experiments. For experiments with iterate averaging, we use the following learning rate schedule instead:

$$\alpha_t = \begin{cases} \alpha_0, & \text{if } \frac{t}{T_{avg}} \leq 0.5 \\ \alpha_0[1 - \frac{(1-\frac{\alpha_{avg}}{\alpha_0})(\frac{t}{T}-0.5)}{0.4}] & \text{if } 0.5 < \frac{t}{T_{avg}} \leq 0.9 \\ \alpha_{avg}, & \text{otherwise} \end{cases} \quad (22)$$

where α_{avg} refers to the (constant) learning rate after iterate averaging activation, and in this paper we set $\alpha_{avg} = \frac{1}{2}\alpha_0$. T_{avg} is the epoch after which iterate averaging is activated, and the methods to determine T_{avg} was described in the main text. This schedule allows us to adjust learning rate smoothly in the epochs leading up to iterate averaging activation through a similar linear decay mechanism in the experiments without iterate averaging, as described above.

D LANCZOS ALGORITHM

In order to empirically analyse properties of modern neural network spectra with tens of millions of parameters $N = \mathcal{O}(10^7)$, we use the Lanczos algorithm (Meurant and Strakoš, 2006), provided for deep learning by Granzio 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,

$$\int_a^b f(\lambda) d\mu(\lambda) = \sum_{j=1}^M \rho_j f(t_j) + R[f] \quad (23)$$

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 normalised eigenvectors of the Lanczos tri-diagonal matrix (Golub and Meurant, 1994). The main properties of the Lanczos algorithm are summarized in the theorems 2,3

Theorem 2. *Let $H^{N \times N}$ be a symmetric matrix with eigenvalues $\lambda_1 \geq \dots \geq \lambda_n$ and corresponding orthonormal eigenvectors z_1, \dots, z_n . If $\mathbf{w}_1 \geq \dots \geq \mathbf{w}_m$ are the eigenvalues of the matrix T_m obtained after m Lanczos steps and q_1, \dots, q_k the corresponding Ritz eigenvectors then*

$$\begin{aligned} \lambda_1 &\geq \mathbf{w}_1 \geq \lambda_1 - \frac{(\lambda_1 - \lambda_n) \tan^2(\mathbf{w}_1)}{(c_{k-1}(1 + 2\rho_1))^2} \\ \lambda_n &\leq \mathbf{w}_k \leq \lambda_m + \frac{(\lambda_1 - \lambda_n) \tan^2(\mathbf{w}_1)}{(c_{k-1}(1 + 2\rho_1))^2} \end{aligned} \quad (24)$$

where c_k is the chebyshev polyomial of order k

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

Theorem 3. *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 normalised eigenvectors of T_k*

Proof: See (Golub and Meurant, 1994). The first term on the RHS of equation 23 using Theorem 3 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

$$\mathbb{E}_v \text{Tr}(v^T H^m v) = \text{Tr} \mathbb{E}_v(v v^T H^m) = \text{Tr}(H^m) = \sum_{i=1}^N \lambda_i = N \int_{\lambda \in \mathcal{D}} \lambda d\mu(\lambda) \quad (25)$$

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 $N \rightarrow \infty$, 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).

E MATHEMATICAL PRELIMINARIES

For an input/output pair $[\mathbf{x} \in \mathbb{R}^{d_x}, \mathbf{y} \in \mathbb{R}^{d_y}]$ and a given model $h(\cdot; \cdot) : \mathbb{R}^{d_x} \times \mathbb{R}^P \rightarrow \mathbb{R}^{d_y}$. Without loss of generality, we consider the family of models functions parameterized by the weight vector \mathbf{w} , i.e., $\mathcal{H} := \{h(\cdot; \mathbf{w}) : \mathbf{w} \in \mathbb{R}^P\}$, with a given loss $\ell(h(\mathbf{x}; \mathbf{w}), \mathbf{y}) : \mathbb{R}^{d_y} \times \mathbb{R}^{d_y} \rightarrow \mathbb{R}$.

The empirical risk (often denote the *loss* in deep learning), its gradient and Hessian are given by

$$R_{emp}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N \ell(h(\mathbf{x}_i; \mathbf{w}), \mathbf{y}_i), \mathbf{g}_{emp}(\mathbf{w}) = \nabla R_{emp}, \mathbf{H}_{emp}(\mathbf{w}) = \nabla^2 R_{emp} \quad (26)$$

The Hessian describes the curvature at that point in weight space \mathbf{w} and hence the risk surface can be studied through the Hessian. By the spectral theorem, we can rewrite

$\mathbf{H}_{emp}(\mathbf{w}) = \sum_{i=1}^P \lambda_i \phi_i \phi_i^T$ in terms of its eigenvalue, eigenvector pairs $[\lambda_i, \phi_i]$. In order to characterise $\mathbf{H}_{emp}(\mathbf{w})$ by a single value, authors typically consider the spectral norm, which is given by the largest eigenvalue of $\mathbf{H}_{emp}(\mathbf{w})$ or the normalised trace, which gives the mean eigenvalue. The Hessian contains P^2 elements, so cannot be stored or eigendecomposed for all but the simplest of models. Stochastic Lanczos Quadrature can be used Meurant and Strakoš (2006), with computational complexity $\mathcal{O}(P)$ to give tight bounds on the extremal eigenvalues and good estimations of $\text{Tr}(\mathbf{H})$ and $\text{Tr}(\mathbf{H}^2)$, along with a moment matched approximation of the spectrum. We use the Deep Learning implementation provided by Granziol et al. (2019). DNNs are typically trained using stochastic gradient descent with momentum, where we iteratively update the weights

$$\begin{aligned} \mathbf{z}_{k+1} &\leftarrow \rho \mathbf{z}_k + \nabla R(\mathbf{w}_k) \\ \mathbf{w}_{k+1} &\leftarrow \mathbf{w}_k - \alpha \mathbf{z}_{k+1} \end{aligned} \tag{27}$$

Where ρ is the momentum. The gradient is usually taken on a randomly selected sub-sample of size $B \ll N$. An epoch is defined as a full training pass of the data, so comprises $\approx N/B$ iterations. Often $L2$ regularisation (also termed weight decay) is added to the loss, which corresponds to $R_{emp}(\mathbf{w}) \rightarrow R_{emp}(\mathbf{w}) + \mu/2 \|\mathbf{w}\|^2$.