NEAR BLOCK-DIAGONALITY IN HESSIAN MATRICES

Anonymous authors
Paper under double-blind review

ABSTRACT

The Hessian matrix of the training loss of a neural network is often approximated by its diagonal blocks, but the reason why this approximation is effective is not known. We prove that the top eigenvector of the Hessian is approximately equal to the weighted concatenation of the eigenvectors of its diagonal blocks provided that the forward projections of the eigenvectors are approximately colinear and their Gram matrix is rank-one. We show experimentally that these conditions hold and discuss why they are to be expected given the structure of the Hessian matrix, providing practical guidance for when the diagonal blocks are an effective approximation.

1 INTRODUCTION

The Hessian matrix of the training loss is a key consideration in neural network training, determining the curvature of the loss landscape. The top eigenvalue of the Hessian, called the **sharpness**, controls the maximum learning rate at which gradient descent is stable and the loss monotonically decreases (3). In classical optimization, advanced solvers such as the Newton method invert the Hessian to compensate for curvature, allowing the solver to take larger steps. However, since the size of the Hessian grows quadratically with the number of network parameters, and modern neural networks have millions to billions of parameters, materializing the Hessian matrix for deep neural networks is computationally intractable. As such, many optimizers instead use approximations of the full Hessian. A number of approaches are based on the assumption that the Hessian is nearly block-diagonal, that is, the second derivatives of the network by parameters in different layers are relatively small (36; 2; 8). This approach to estimating the Hessian has been known for some time (4), but the reasons for its success are unknown.

In this work, we show why the diagonal blocks of the Hessian matrix yield an accurate approximation for the top eigenvector. The Hessian matrix can be decomposed into the sum of three components: G, Γ, and V (20; 21). G contains the second partial derivatives of the criterion, Γ contains the second partial derivatives of the activation functions, and V contains the quadratic interactions of the parameters. The Hessian is dominated by G, and both G and Γ can be interpreted as second-moment matrices (30), where a random matrix projects a vector into another space and then back to the original space. We prove a Lemma bounding the difference between the sharpness of a second moment matrix of this form and the sum of the sharpnesses of its diagonal blocks. We show that the tightness of the bound is controlled by the principal values of the projected vectors, and experimentally validate that these principal values are highly concentrated in practice. This implies that the top eigenvector of the Hessian matrix is extremely well-approximated by the weighted concatenation of the top eigenvectors of its diagonal blocks. However, the lower eigenvectors are not.

In Section 2, we establish notation and review background on the Hessian matrix of the training loss of neural networks. In Section 3, we discuss our mathematical methodology and Lemmas. In Section 4, we lay out experiments to empirically verify the terms in our approximation. In Section 5, we discuss the results. In Section 6, we conclude.

2 BACKGROUND AND NOTATION

We define a neural network to be the composition of a sequence of alternating linear layers and activation layers. A linear layer is a linear transformation:

$$\hat{x}^i = \alpha^i(\theta^i)x^{i-1}$$

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Where x^i are the network **activations**, $\hat{x}^i$ are the network **preactivations**, θ^i are the **parameters** of layer i, and α^i is an affine map of the form:

$$\alpha^i(\theta^i)_{j,k} = 0 \text{ or } 1 \text{ or } \theta^i_m \text{ for some } m$$

And the next layer's activations are:

$$x^i = \sigma^i(\hat{x}^i)$$

For some parameter-less **activation function** σ^i. The network inputs are treated as the 0th-layer activations x^0, and the network outputs as the n_Lth-layer preactivations $z = \hat{x}^{n_L}$, where n_L is the number of layers. We adopt the convention that σ^i appends a constant 1 to the bottom of every activation, which removes the need for biases as a separate term. This definition of a neural network includes multilayer perceptrons, convolutional neural networks, and transformers.

We define $l(z, y)$ to be the **criterion** measuring the difference between the network outputs and the labels y. We define the **training loss** to be the expectation of the criterion over the network inputs:

$$\widetilde{\mathcal{L}}(\theta) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{T}} l(z(x,\theta), y)$$

Where $\mathcal{T}$ is the **training set**. Note that all expectations are taken with regard to the training data. We next make the following assumption:

Assumption 2.1. *The criterion $l(z, y)$ is convex as a function of z.*

Assumption 2.1 holds for cross-entropy, mean squared error, and most other commonly-used criteria.

The **Hessian matrix** is the matrix of second derivatives of the training loss $\widetilde{\mathcal{L}}(\theta)$. Since the number of entries of the Hessian is equal to the square of the number of parameters, it is not feasible to materialize the Hessian for modern neural networks. Instead, current approaches rely on computing Hessian-vector products using automatic differentiation (26) and using either power iteration or the Lanczos method to calculate eigenvalues and eigenvectors(24; 1).

The Hessian is commonly decomposed into the sum of two terms, G and H (17):

$$\frac{\partial^2 \widetilde{\mathcal{L}}}{\partial \theta^2} = G + H$$

$$G = \mathbb{E}\left(\left(\frac{\partial z}{\partial \theta}\right)^T \frac{\partial^2 l}{\partial z^2}\frac{\partial z}{\partial \theta}\right)$$

$$H = \mathbb{E}\left(\sum_i \frac{\partial l}{\partial z_i}\frac{\partial^2 z_i}{\partial \theta^2}\right)$$

G corresponds to the second partial derivatives of the criterion, while H corresponds to the second partial derivatives of the network. G is often referred to as the **Gauss-Newton matrix** (30) or **outer product matrix** (32). H has been variously referred to as the **nonlinear modeling error matrix** (5), the **residual Hessian** (20), or the **functional Hessian** (32). In practice, G consistently dominates the Hessian matrix.

H can also be further decomposed into the sum of two more matrices, Γ and V (20; 21):

$$H = \Gamma + V$$

$$\Gamma = \sum_i \sum_j \mathbb{E}\left(\left(\frac{\partial \hat{x}^i}{\partial \theta}\right)^T \left(\frac{\partial l}{\partial x^i_j}\frac{\partial^2 x^i_j}{\partial (\hat{x}^i)^2}\right)\frac{\partial \hat{x}^i}{\partial \theta}\right)$$

$$V = \sum_{i,j,k} \mathbb{E}\left(\frac{\partial l}{\partial z_i}\frac{\partial z_i}{\partial \hat{x}^j}\frac{\partial \hat{x}^j}{\partial \theta}\frac{\partial \hat{x}^j}{\partial \hat{x}^k}\frac{\partial \hat{x}^k}{\partial \theta}\right)$$

Γ corresponds to the second partial derivatives of the activation functions, while V corresponds to the quadratic interactions of the network parameters.

The Hessian matrix was observed to be approximately block-diagonal quite early (4), but explanations of the cause remain scarce. Nor has it gone uncontested: (10) notably claimed that near-block-diagonality breaks down for large transformer models. However, their experiments only showed that

the block diagonal no longer accurately approximates Hessian-vector products for random vectors. Given the high dimension of the parameter space, this does not rule out that the block diagonal might be a good approximation for the Hessian-vector products of *particular* vectors, such as the top eigenvectors. Recent work has found analytic explanations for near-block-diagonality in some special cases, but is limited to linear networks and one-layer networks (7). Despite this, near-block-diagonality has been heavily exploited in the development of new optimizers (36; 2; 8).

3 METHODOLOGY

3.1 DECOMPOSITION OF G

Recall that G is:

$$G = \mathbb{E}\left(\left(\frac{\partial z}{\partial \theta}\right)^T \frac{\partial^2 l}{\partial z^2}\frac{\partial z}{\partial \theta}\right)$$

Finding the sharpness of G is therefore equivalent to:

$$\lambda^1(G) = \max_{||v||=1} \mathbb{E}\left(v^T\left(\frac{\partial z}{\partial \theta}\right)^T \frac{\partial^2 l}{\partial z^2}\frac{\partial z}{\partial \theta}v\right) \tag{1}$$

Recall that, in Assumption 2.1, we assumed that l is convex. This implies that its Hessian matrix is positive semi-definite, and that it therefore has a unique real positive semi-definite square root, which we denote R. We can then rewrite Equation 1 to express G as a second-moment matrix (30):

$$\lambda^1(G) = \max_{||v||=1} \mathbb{E}\left|\left|R\frac{\partial z}{\partial \theta}v\right|\right|^2 \tag{2}$$

From the chain rule and the definition of α^i, we obtain:

$$\frac{\partial z_m}{\partial \theta^i_j} = \sum_{k,k' \in I_j^{\alpha^i}} \frac{\partial z_m}{\partial \hat{x}^i_{k'}} x^{i-1}_k$$

$$I_j^{\alpha^i} = \left\{(k,k') : \alpha^i(\theta^i)_{k,k'} = \theta^i_j\right\}$$

This allows us to rewrite Equation 2 as:

$$\lambda^1(G) = \max_{||v||=1} \mathbb{E}\left|\left|\sum_{i=1}^{n_L} R\frac{\partial z}{\partial \hat{x}^i}\hat{\alpha}^i(v_{\theta^i})x^{i-1}\right|\right|^2 \tag{3}$$

Where we use v_{θ^i} to denote the block of v corresponding to the parameters θ^i, and we define $\hat{\alpha}^i$ to be:

$$\hat{\alpha}^i(\theta)_{j,k} = \left\{ \begin{array}{ll} \alpha^i(\theta^i)_{j,k} & \alpha^i(\theta)_{j,k} = \theta_r \text{ for some } r \\ 0 & \alpha^i(\theta)_{j,k} = 0 \text{ or } 1 \end{array} \right\}$$

We then make use of the following Lemmas:

Lemma 3.1. *Given random matrices M_i, random vectors U_i, and linear functions $\hat{\alpha}^i$, we define W_i' to be the vectors that achieve the maximum of:*

$$\max_{||W_1||^2+\ldots||W_{n_L}||^2=1} \mathbb{E}\left|\left|\sum_i M_i\hat{\alpha}^i(W_i)U_i\right|\right|^2$$

Then:

$$\max_{||W_1||^2+\ldots||W_{n_L}||^2=1} \mathbb{E}\left|\left|\sum_i M_i\hat{\alpha}^i(W_i)U_i\right|\right|^2 = \rho(M,U)\sum_i \mathbb{E}\left|\left|M_i\hat{\alpha}^i\left(\frac{W_i'}{||W_i'||}\right)U_i\right|\right|^2$$

Where:

$$\rho(M,U) = \frac{\lambda^1\left(\mathbb{E}(K')^TK'\right)}{\sum_j \lambda^j\left(\mathbb{E}(K')^TK'\right)}$$

And K' is the matrix whose columns are:

$$K_i' = M_i\hat{\alpha}^i\left(\frac{W_i'}{||W_i'||}\right)U_i$$

Proof. See Section 7.1 in the Supplemental Material. □

Note that, since $\mathbb{E}K^TK$ is positive-semidefinite, $0 \leq \rho(M,U) \leq 1$. For notational convenience, we define:

$$\rho(G) = \rho\left(R\frac{\partial z}{\partial \hat{x}^i}, x^{i-1}\right)$$

Lemma 3.2. *Given random matrices M_i and random vectors U_i, we define W_i' to be the vectors that achieve the maximum of:*

$$\max_{||W_1||^2+...||W_{n_L}||^2=1} \mathbb{E}\left|\left|\sum_i M_i\hat{\alpha}^i(W_i)U_i\right|\right|^2$$

And we define:

$$\hat{W}_i = \underset{||W_i||=1}{\operatorname{argmax}} \mathbb{E}\left|\left|M_i\hat{\alpha}^i\left(W_i\right)U_i\right|\right|^2$$

$$K_i = M_i\alpha^i\left(\hat{W}_i\right)U_i$$

If there exists a random vector K_0 and random scalars k_i such that $||K_0|| = 1, K_i = k_iK_0$, then:

$$\hat{W}_i = \frac{W_i'}{||W_i'||}$$

Proof. See Section 7.1 in the Supplemental Material. □

From these lemmas, we obtain:

Corollary 3.1. *For any neural network:*

$$\lambda^1(G) \leq \sum_i \max_{||v_i||=1} \mathbb{E}\left|\left|R\frac{\partial z}{\partial \hat{x}^i}\hat{\alpha}^i(v_i)x^{i-1}\right|\right|^2 \tag{4}$$

In addition, let v_i' be the v_i that achieve the maximum in Equation 4, and define:

$$Z_i = R\frac{\partial z}{\partial \hat{x}^i}\hat{\alpha}^i(v_i')x^{i-1}$$

And let Z be the random matrix whose columns are Z_i. Equality holds in Equation 4 if $\mathbb{E}Z^TZ$ is rank one and there exists a random vector Z_0 and random scalars z_i' such that $Z_i = z_i'Z_0$.

Proof. Equation 4 is immediate from applying Lemma 3.1 to Equation 3, and observing that:

$$\max_{||v_i||=1} \mathbb{E}\left|\left|R\frac{\partial z}{\partial \hat{x}^i}\hat{\alpha}^i(v_i)x^{i-1}\right|\right|^2 \geq \mathbb{E}\left|\left|R\frac{\partial z}{\partial \hat{x}^i}\hat{\alpha}^i(v_i')x^{i-1}\right|\right|^2$$

For any v_i' such that $||v_i'|| = 1$. The conditions for equality follow from Lemma 3.2 and the fact that, first, if $\mathbb{E}Z^TZ$ is rank one, then $\lambda^2(\mathbb{E}Z^TZ) = ... = 0$, which implies $\rho(G) = 1$. □

We refer to the summands of Corollary 3.1 as the **layerwise G sharpnesses**. Each of these summands correspond to the sharpness of one of the diagonal blocks of G.

Let G_i be the ith diagonal block of G. Then G_i is equal to (19; 35):

$$G_i = \mathbb{E}D_i^TD_i$$

$$D_i = R\frac{\partial z}{\partial \hat{x}^i} \otimes x^{i-1}$$

Where $\otimes$ denotes the Kronecker product. If we multiply $D_i^TD_i$ by some vector v, D_i first projects that vector to a random vector in the output space, then D_i^T projects it back to the original space. When v is the top eigenvector of G_i, then $Z_i = D_iv_i'$. Corollary 3.1 says that the sum of the sharpnesses of the diagonal blocks will be a good approximator for the sharpness of G provided that:

- For any given sample of D_i, Z_i are colinear, that is, $Z_i = z_i' Z_0$ for some random scalars z_i'. Note that Z_0 can vary depending on the network input, so this does not imply that *all* Z_i are colinear, only that Z_i, Z_j *for the same input* are colinear.
- The norms $||Z_i||$ vary with each other colinearly, that is, $\mathbb{E}Z^T Z$ is rank one. So if Z_i is unusually large for some particular input, Z_j will also be unusually large for that input, and vice-versa.

This also implies that, if equality holds in Corollary 3.1, then the top eigenvector of G is equal to the weighted concatenation:

$$(v^1(G))_{\theta^i} = c_i v_i'$$

For some scalar constants c_i.

3.2 Decomposition of Γ

Recall that Γ is:

$$\Gamma = \sum_i \sum_j \mathbb{E}\left(\left(\frac{\partial \hat{x}^i}{\partial \theta}\right)^T \left(\frac{\partial l}{\partial x_j^i}\frac{\partial^2 x_j^i}{\partial (\hat{x}^i)^2}\right)\frac{\partial \hat{x}^i}{\partial \theta}\right)$$

We define the i summands of Γ to be Γ^i, and proceed analogously to our decomposition of G:

$$\lambda^1(\Gamma^i) = \max_{||v||=1} \mathbb{E}\left(v^T \left(\frac{\partial \hat{x}^i}{\partial \theta}\right)^T Q^i \frac{\partial \hat{x}^i}{\partial \theta} v\right)$$

$$Q^i_{j,k} = \sum_m \frac{\partial l}{\partial x_m^i}\frac{\partial^2 x_m^i}{\partial \hat{x}_j^i \partial \hat{x}_k^i}$$

In the case of an elementwise activation function, Q^i is a diagonal random matrix equal to:

$$Q^i = \text{diag}\left(\frac{\partial l}{\partial x^i} \odot \sigma''(\hat{x}^i)\right)$$

Where $\odot$ denotes the elementwise product. As a result, Q^i will generally not be positive semi-definite. But even when the Hessian matrix does not have a real matrix square root, it will have a *complex* matrix square root. This gives us a bound on the eigenvalues of Γ^i:

$$\lambda^1(\Gamma^i) \leq \max_{||v||=1} \mathbb{E}\left|\left|\sqrt{Q^i}\frac{\partial \hat{x}^i}{\partial \theta} v\right|\right|^2$$

Where $\sqrt{Q^i}$ is the matrix square root of Q^i. This in turn gives us:

$$\lambda^1(\Gamma^i) \leq \max_{||v||=1} \mathbb{E}\left|\left|\sum_{k=1}^{i} \sqrt{Q^i}\frac{\partial \hat{x}^i}{\partial \hat{x}^k}\hat{\alpha}^k(v_{\theta^k})x^{k-1}\right|\right|^2 \tag{5}$$

Analogous to $\rho(G)$, we define:

$$\rho(\Gamma^i) = \rho\left(\sqrt{Q^i}\frac{\partial \hat{x}^i}{\partial \hat{x}^k}, x^{k-1}\right)$$

Then:

Corollary 3.2.

$$\lambda^1(\Gamma^i) \leq \sum_k \max_{||v_k||=1} \mathbb{E}\left|\left|\sqrt{Q^i}\frac{\partial \hat{x}^i}{\partial \hat{x}^k}\hat{\alpha}^k(v_k)x^{k-1}\right|\right|^2$$

Proof. This is immediate from applying Lemma 3.1 to Equation 5 and applying the same logic used to prove Corollary 3.1. □

We refer to the summands of Corollary 3.2 as the **layerwise Γ^i sharpnesses**.

4 EXPERIMENTS

We perform two sets of experiments to determine the tightness of the bounds provided by Corollaries 3.1 and 3.2. The first set of experiments uses small multilayer perceptrons (MLPs) so that we can track the evolution of the sharpness, $\rho(G), \rho(\Gamma^i)$, and the component ratios over the course of training. The second uses already-trained, large networks such as CNNs to validate that these results also apply at scale.

In the first set of experiments, we train MLPs on two datasets: a randomly-selected subset of 1,024 images from CIFAR-10 (13) using cross-entropy, and a randomly-selected subset of 1,024 samples from the ECL time series dataset (15) using mean squared error. We follow (23) in treating ECL as many separate univariate time series instead of a single multivariate time series, use 96 as the lookback, and predict one step ahead. We vary the the width of the network (256, 512, 1024), the depth of the network (4, 6, 8), and the activation function (tanh, softplus with $\beta = 10$ (22)). We initialize each network using the Kaiming initialization (11), using $5/3$ as the gain for the tanh and $\sqrt{2}$ as the gain for the softplus. For each combination of hyperparameters we train 16 networks, using four choices of random seed to select the data subset and four choices of random seed to initialize the network. We train until loss drops below 0.2 using full-batch gradient descent with learning rate $\eta = 0.01$ (on CIFAR-10) or $\eta = 0.001$ (on ECL) and no momentum.

Every ten iterations, we calculate:

- The Hessian, G, and Γ^i sharpnesses,
- The G and Γ^i layerwise sharpnesses,
- $\rho(G)$ and $\rho(\Gamma^i)$,
- The ratio:
$$\frac{\sum_i \mathbb{E}\left|\left|R\frac{\partial z}{\partial \hat{x}^i}\hat{\alpha}^i(v_{\theta^i})x^{i-1}\right|\right|^2}{\sum_i \mathbb{E}\left|\left|R\frac{\partial z}{\partial \hat{x}^i}\hat{\alpha}^i(v_i')x^{i-1}\right|\right|^2}$$
Where v is the top eigenvector of G and v_i' is the argmax of Corollary 3.1. We refer to this as the G **component ratio**.
- The ratio:
$$\frac{\sum_k \mathbb{E}\left|\left|\sqrt{Q^i}\frac{\partial \hat{x}^i}{\partial \hat{x}^k}\hat{\alpha}^k(v_{\theta^k})x^{k-1}\right|\right|^2}{\sum_k \mathbb{E}\left|\left|\sqrt{Q^i}\frac{\partial \hat{x}^i}{\partial \hat{x}^k}\hat{\alpha}^k(v_k')x^{k-1}\right|\right|^2}$$
Where v is the top eigenvector of Γ^i and v_k' is the argmax of Corollary 3.2 for Γ^i. We refer to these as the Γ^i **component ratios**.

See Section 7.2 in the Supplemental Material for implementation details of these calculations. Note that, for very small eigenvalues - which occurs for some Γ^i - estimation errors become significant, and power iteration fails to converge. We exclude iterations where power iteration failed to converge from subsequent calculations.

In the second set of experiments, we analyze large, pre-trained models sourced from common repositories to validate that our conclusions remain valid for large, deep networks. Specifically, we analyze:

- A VGG-16 without batch normalization trained on ImageNet (31; 6).
- A ResNet-18 trained on ImageNet (12; 6).

Since the contribution of Γ to the sharpness is relatively small, we collect only the values for G for the large models.

5 RESULTS AND DISCUSSION

We plot the sharpness of the Hessian, G, and Γ^i for a representative experiment using width 512, depth 4, softplus activation, and CIFAR-10 dataset as the solid lines in Figure 1. We also plot the upper bound established by Corollaries 3.1 and 3.2 as the dashed lines.

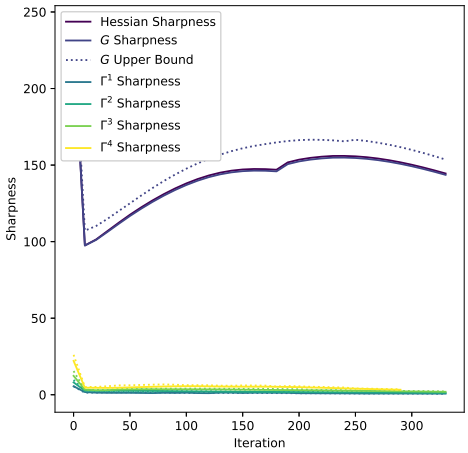

Figure 1: The sharpness and upper bound for G and Γ^i for an example MLP with depth 4, width 1024, and softplus activation trained on CIFAR-10

Consistent with previous results, the sharpness of the Hessian matrix as a whole is almost always very close to the sharpness of G, to the point that it is difficult to see in Figure 1. The median ratio between the sharpness of the Hessian and the sharpness of G is $\times 1.02$. In 93.6% of experiments the ratio never exceeds $\times 1.25$, and over all experiments the ratio never exceeds $\times 1.84$[1].

Across all of our MLP experiments, the G upper bound is never more than $\times 1.42$ greater than the actual value for G, and its median value is $\times 1.05$. The upper bound can be significantly looser for Γ^i, up to $\times 7.74$, but its median value is only $\times 1.07$.

$\mathbb{E}K^T K$ is nearly-colinear for both G and Γ^i in all experiments. For all MLPs, $\rho(G)$ remains between 0.85 and 1.00, with median 0.97, and $\rho(\Gamma^i)$ between 0.80 and 1.00, with median 0.99. We plot the values for our representative experiment in the left subplot of Figure 2. Although in our example the $\rho(G)$ increases over time while $\rho(\Gamma^i)$ decrease, this is not consistent between experiments, and there is no obvious temporal pattern.

Similarly, the eigenvectors of the block diagonals are always close to the components of the eigenvector of the full matrix. The G component ratio always remains between 0.76 and 1.0, with median 0.97. The Γ^i component ratios have some extreme outliers, varying between 0.02 and 1.0, but the median is 0.94. We plot the values for our representative experiment in the right subplot of Figure 2. Again, there is no obvious temporal pattern. The Γ^i component ratios do exhibit significantly greater temporal variance, especially Γ^1; this is likely because, for Γ^i early in the network, the denominator of the ratio can be quite small, so that small estimator errors can cause significant swings.

We display the minimum and maximum values of $\rho(G), \rho(\Gamma^i)$, the G component ratio, and the Γ^i component ratios for every individual choice of parameters in Table 1. There is no obvious dependence of the values on any individual hyperparameter.

We find similar results for our large, pre-trained networks. We display the values of $\rho(G)$ and the G component ratio in Table 2. In all cases $\rho(G)$ remains well above 0.5 and the G ratio remains above 0.75.

These results should not surprise us. Recall that the requirements for equality in Corollary 3.1 are that $\mathbb{E}Z^T Z$ is rank one and the Z_i are colinear, where Z_i are defined as:

$$Z_i = R\frac{\partial z}{\partial \hat{x}^i}\hat{\alpha}^i(v_i')x^{i-1}$$

[1]We do observe a small number of experiments where the ratio of the Hessian sharpness to the G sharpness is unusually high, exceeding $\times 1.5$. All used CIFAR-10, tanh activation, depth 8, and width 256. We leave exploration of these cases to future work.

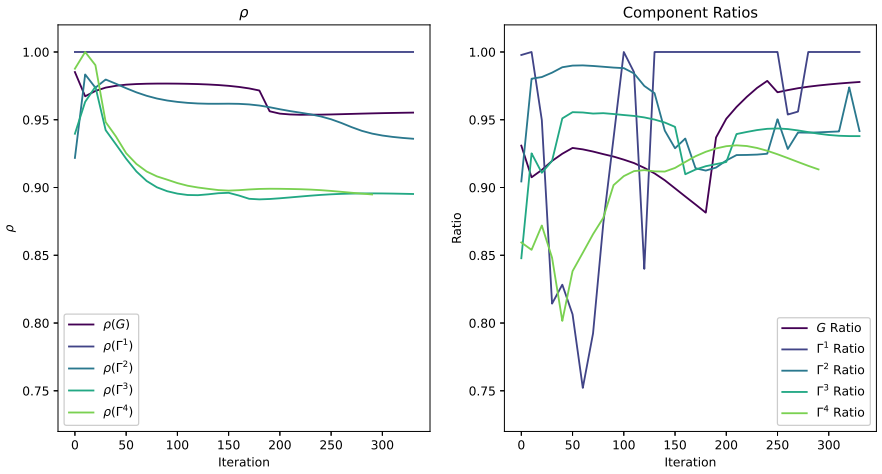

Figure 2: The $\rho(G)$ and $\rho(\Gamma^i)$ for an example MLP with depth 4, width 1024, and softplus activation trained on CIFAR-10

	$\rho(G)$		G Ratio		$\rho(\Gamma)$		Γ Ratio	
Factor	Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.
activation=softplus	0.85	1.00	0.76	1.00	0.80	1.00	0.02	1.00
activation=tanh	0.86	1.00	0.84	1.00	0.88	1.00	0.02	1.00
depth=4	0.85	1.00	0.78	1.00	0.81	1.00	0.02	1.00
depth=6	0.87	1.00	0.81	1.00	0.81	1.00	0.02	1.00
depth=8	0.85	1.00	0.76	1.00	0.80	1.00	0.02	1.00
task=cifar	0.85	0.99	0.76	1.00	0.80	1.00	0.02	1.00
task=electricity	0.96	1.00	0.84	1.00	0.88	1.00	0.02	1.00
width=1024	0.93	1.00	0.76	1.00	0.84	1.00	0.02	1.00
width=256	0.85	0.99	0.80	1.00	0.80	1.00	0.04	1.00
width=512	0.87	1.00	0.78	1.00	0.81	1.00	0.02	1.00

Table 1: Breakdown of minimum and maximum values by different hyperparameters.

Model	$\rho(G)$	G Ratio
VGG-16	0.85	0.87
ResNet-18	0.65	0.77

Table 2: Results of large model experiments

Where the Jacobian is:

$$\frac{\partial z}{\partial \hat{x}^i} = \hat{\alpha}^{n_L}(\theta^{n_L})(\sigma')^{n_L-1}(\hat{x}^{n_L-1})\hat{\alpha}^{n_L-1}(\theta^{n_L-1})...(\sigma')^i(\hat{x}^i)\hat{\alpha}^i(\theta^i)$$

Both Z_i and Z_{i-1} are the product of $R\frac{\partial z}{\partial \hat{x}^i}$ and a random vector, $\hat{\alpha}^i(v_i')x^{i-1}$ and $\frac{\partial \hat{x}^i}{\partial \hat{x}^{i-1}}\hat{\alpha}^{i-1}(v_{i-1}')x^{i-2}$, where v_i', v_{i-1}' are chosen to maximize $||Z_i||, ||Z_{i-1}||$. Each sample of Z_i and Z_{i-1} will therefore tend to be close to parallel to the top left singular vector of $R\frac{\partial z}{\partial \hat{x}^i}$, and hence to each other. Thus, it should not surprise us that Z_i, Z_{i-1} are colinear.

Furthermore, if Z_i, Z_j *are* approximately colinear for any given input, then:

$$(\mathbb{E}Z^TZ)_{i,j} \approx \mathbb{E}\left(||Z_i|| \cdot ||Z_j||\right)$$

This will be rank 1 if the ratios $||Z_i||/||Z_j||$ are approximately independent of the network input. From the definition of Z_i, this should not surprise us either: $||Z_i||$ will primarily be controlled by the singular values of $\hat{\alpha}^k(\theta^k), \alpha^k(\theta^k)$. The activation functions - the only components that vary with network input - act analogously to noise terms. Analogous reasoning applies to Γ^i as well. Therefore, it should not surprise us if Corollary 3.1 and 3.2 are close to equality.

When the top eigenvalue is approximately equal to the sum of the top eigenvalues for each of the diagonal blocks, then for every layer of the network, the parameters of the top eigenvector of G corresponding to that layer will be equal to a scalar multiplied by the top eigenvector of the block diagonal of G. Since G dominates the Hessian, the top eigenvector of the Hessian is approximately equal to the weighted concatenation of the top eigenvectors of the block diagonals.

However, this result is distinct from the top eigenvector or eigenvalue of the Hessian being equal to the top eigenvector or eigenvalues of the block diagonal approximation of the Hessian:

Lemma 5.1. *Let M be a block-diagonal matrix with blocks M_i, so that:*

$$M = \begin{pmatrix} M_1 & & \\ & ... & \\ & & M_{n_L} \end{pmatrix}$$

Let $\hat{M}$ be the M_i with the largest eigenvalue. Then $\lambda^1(M) = \lambda^1(\hat{M}), v^1(M) = v^1(\hat{M})$.

Proof. See Section 7.1 in the Supplemental Material. □

Instead, the top eigenvalue of G will be approximately equal to the *sum* of the top eigenvalues of its diagonal blocks.

Furthermore, this result tells us nothing about the lower eigenvectors. The eigenvectors and eigenvalues of the Hessian separate into a small "top" subspace and larger "bulk" subspace (27; 28; 29; 24; 25). Although we did not test this in our experiments, we would expect approximations similar to Corollaries 3.1 and 3.2 to hold for other eigenvectors in the top subspace, but with the approximation becoming less accurate the further down one gets, until completely failing in the bulk subspace. (10) found that, for randomly chosen unit vectors, the block diagonals do not provide a good approximation of the Hessian-vector product. Since a randomly chosen vector will be approximately perpendicular to the top eigenvector with high probability, these results are consistent with ours.

As such, whether the diagonal blocks will be an acceptable approximation for the Hessian in applications such as optimization depends on what the Hessian is being used for. If only the top eigenvector is needed, the diagonal blocks, used appropriately, are highly accurate. If eigenvectors or eigenvalues outside the top are needed, then we expect accuracy to be poor.

6 Conclusion

Our results explain why the block diagonals are often, but not always, an accurate approximation of the Hessian matrix. We provide a bound on the sharpness of G and Γ^i in terms of their block diagonals and show that this bound is tight in practice and that this result is to be expected. If only the top eigenvector is needed, the block diagonal is a good approximation, but not if eigenvalues or lower eigenvectors are needed.

REFERENCES

[1] Ghorbani Behrooz, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. *International Conference on Machine Learning*, 2019.

[2] Aleksandar Botev, Hippolyt Ritter, and David Barber. Practical Gauss-Newton optimisation for deep learning. In *International Conference on Machine Learning*, 2017.

[3] Jeremy M. Cohen, Simran Kaur, Yuanzhi Li, J. Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*, 2021.

[4] Ronan Collobert. *Large Scale Machine Learning*. PhD thesis, Pierre and Marie Curie University, 2004.

[5] Yann Dauphin, Atish Agarwala, and Hossein Mobahi. Neglected Hessian component explains mysteries in sharpness regularization. *Advances in Neural Information Processing Systems*, 38, 2024.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[7] Zhaorui Dong, Yushun Zhang, Zhi-Quan Luo, Jianfeng Yao, and Ruoyu Sun. Towards quantifying the Hessian structure of neural networks, 2025.

[8] Donald Goldfarb, Yi Ren, and Achraf Bahamou. Practical quasi-Newton methods for training deep neural networks. In *Neural Information Processing Systems*, 2020.

[9] Noah Golmant, Zhewei Yao, Amir Gholami, Michael Mahoney, and Joseph Gonzalez. pytorch-hessian-eigenthings: efficient PyTorch Hessian eigendecomposition, 2018.

[10] Diego Granziol and Khurshid Juarev. Hessian spectral analysis at foundation model scale, 2026.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, page 1026–1034, 2015.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Computer Vision and Pattern Recognition*, 2016.

[13] Alex Krizhevsky. Learning multiple layers of features from tiny images. `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`, 2009.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84—90, 2017.

[15] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. In *SIGIR '18: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 95–104, 2018.

[16] Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45(4):255–282, 1950.

[17] Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. *Efficient BackProp*, pages 9–48. Lecture Notes in Computer Science LNCS-1524. Springer Verlag, 1998.

[18] Nenad Markus. Fusing batch normalization and convolution in runtime, 2018.

[19] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International Conference on Machine Learning*, 2015.

[20] Eiji Mizutani and Stuart Dreyfus. On practical use of stagewise second-order backpropagation for multi-stage neural-network learning. In *IEEE International Joint Conference on Neural Networks Proceedings*, pages 1302–1307, 2007.

[21] Eiji Mizutani and Stuart Dreyfus. Second-order stagewise backpropagation for Hessian-matrix analyses and investigation of negative curvature. *Neural Networks*, 21:193–203, 2008.

[22] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 807—814, 2010.

[23] Yuqi Nie, Nam Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *International Conference on Learning Representations*, 2023.

[24] Vardan Papyan. The full spectrum of deepnet Hessians at scale: Dynamics with SGD training and sample size, 2018.

[25] Vardan Papyan. Measurements of three-level hierarchical structure in the outliers in the spectrum of deepnet Hessians. *International Conference on Machine Learning*, 2019.

[26] Barak A. Pearlmutter. Fast exact multiplication by the Hessian. *Neural Computation*, 6(1):147—-160, 1994.

[27] S. Saarinen, R. Bramley, and G. Cybenko. Ill-conditioning in neural network training problems. *SIAM Journal on Scientific Computing*, 14(3):693–714, 1993.

[28] Levent Sagun, Leon Bottou, and Yann LeCun. Eigenvalues of the Hessian in deep learning: Singularity and beyond, 2017.

[29] Levent Sagun, Utku Evci, V. Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the Hessian of over-parameterized neural networks. *International Conference on Learning Representations*, 2018.

[30] Nicol Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14:1723–1738, 2002.

[31] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. 2015.

[32] Sidak Pal Singh, Gregor Bachmann, and Thomas Hofmann. Analytic insights into structure and rank of neural network Hessian maps. *Neural Information Processing Systems*, 2021.

[33] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[34] Richard von Mises and Hilda Pollaczek-Geiringer. Praktische verfahren der gleichungsauflösung . *Zamm-zeitschrift Fur Angewandte Mathematik Und Mechanik*, 9:152–164, 1929.

[35] Yikai Wu, Xingyu Zhu, Chenwei Wu, Annie Wang, and Rong Ge. Dissecting Hessian: Understanding common structure of hessian in neural networks, 2022.

[36] Huishuai Zhang, Caiming Xiong, James Bradbury, and Richard Socher. Block-diagonal hessian-free optimization for training neural networks, 2017.

7 SUPPLEMENTAL MATERIAL

7.1 DEFERRED PROOFS

Lemma 3.1. *Given random matrices M_i, random vectors U_i, and linear functions $\hat{\alpha}^i$, we define W_i' to be the vectors that achieve the maximum of:*

$$\max_{||W_1||^2+...||W_{n_L}||^2=1} \mathbb{E}\left|\left|\sum_i M_i\hat{\alpha}^i(W_i)U_i\right|\right|^2$$

Then:

$$\max_{||W_1||^2+...||W_{n_L}||^2=1} \mathbb{E}\left|\left|\sum_i M_i\hat{\alpha}^i(W_i)U_i\right|\right|^2 = \rho(M,U)\sum_i \mathbb{E}\left|\left|M_i\hat{\alpha}^i\left(\frac{W_i'}{||W_i'||}\right)U_i\right|\right|^2$$

Where:

$$\rho(M,U) = \frac{\lambda^1\left(\mathbb{E}(K')^TK'\right)}{\sum_j \lambda^j\left(\mathbb{E}(K')^TK'\right)}$$

And K' is the matrix whose columns are:

$$K_i' = M_i\hat{\alpha}^i\left(\frac{W_i'}{||W_i'||}\right)U_i$$

Proof. Let W_i' be the vectors that achieve the maximum of:

$$\max_{||W_1||^2+...||W_{n_L}||^2=1} \mathbb{E}\left|\left|\sum_i M_i\hat{\alpha}^i(W_i)U_i\right|\right|^2$$

Define the vector d by $W_i' = d_i\hat{W}_i'$, where $||\hat{W}_i'|| = 1$. Since $||W_1'||^2 + ... = 1$ and $||\hat{W}_i'|| = 1$, we know that $||d|| = 1$. Define K_i' as:

$$K_i' = M_i\alpha^i(\hat{W}_i')U_i$$

And define K' to be the matrix whose ith column is K_i'. Since $\hat{\alpha}_i$ is linear, this allows us to rewrite the above as:

$$\max_{||W_1||^2+||W_2||^2+...=1} \mathbb{E}\left|\left|\sum_i M_i\hat{\alpha}^i(W_i)U_i\right|\right|^2 = \max_{||d||=||\hat{W}_1'||=...=1} \mathbb{E}\left|\left|\sum_i d_iM_i\hat{\alpha}^i(\hat{W}_i')U_i\right|\right|^2 =$$

$$\max_{||d||=...=1} \mathbb{E}||K'd||^2 = \max_{||d||=...=1} d^T\left(\mathbb{E}(K')^TK'\right)d$$

Therefore d is the top eigenvector of $\mathbb{E}(K')^TK'$, giving us:

$$\max_{||W_1||^2+||W_2||^2+...=1} \mathbb{E}\left|\left|\sum_i M_i\hat{\alpha}^i(W_i)U_i\right|\right|^2 = \lambda^1\left(\mathbb{E}(K')^TK'\right)$$

Now consider the right-hand term. We observe that, since $\hat{W}_i$ realize the maximum:

$$\sum_i \mathbb{E}\left|\left|M_i\hat{\alpha}^i(\hat{W}_i)U_i\right|\right|^2 \geq \sum_i \mathbb{E}\left|\left|M_i\hat{\alpha}^i(\hat{W}_i')U_i\right|\right|^2$$

And we can rewrite it in terms of K' as well:

$$\sum_i \mathbb{E}\left|\left|M_i\hat{\alpha}^i(\hat{W}_i')U_i\right|\right|^2 = \sum_i \mathbb{E}\,||K_i'||^2 = \sum_{i,j}\mathbb{E}(K_{i,j}')^2 = \text{Tr}\,\left(\mathbb{E}(K')^TK'\right) = \sum_i \lambda^i\left(\mathbb{E}(K'^TK'\right)$$

$(K')^TK'$ is a positive semi-definite symmetric matrix, so $\mathbb{E}(K')^TK'$ is also a positive semi-definite symmetric matrix. Therefore, its eigenvalues are real and non-negative. Therefore:

$$\max_{||W_1||^2+...||W_{n_L}||^2=1} \mathbb{E}\left|\left|\sum_i M_i\hat{\alpha}^i(W_i)U_i\right|\right|^2 \leq \sum_i \mathbb{E}\left|\left|M_i\hat{\alpha}^i(\hat{W}_i)U_i\right|\right|^2$$

□

Lemma 3.2. *Given random matrices M_i and random vectors U_i, we define W_i' to be the vectors that achieve the maximum of:*

$$\max_{||W_1||^2+...||W_{n_L}||^2=1} \mathbb{E}\left|\left|\sum_i M_i\hat{\alpha}^i(W_i)U_i\right|\right|^2$$

And we define:

$$\hat{W}_i = \underset{||W_i||=1}{\operatorname{argmax}} \mathbb{E}\left|\left|M_i\hat{\alpha}^i\left(W_i\right)U_i\right|\right|^2$$

$$K_i = M_i\alpha^i\left(\hat{W}_i\right)U_i$$

If there exists a random vector K_0 and random scalars k_i such that $||K_0|| = 1, K_i = k_iK_0$, then:

$$\hat{W}_i = \frac{W_i'}{||W_i'||}$$

Proof. Define:

$$K_i' = M_i\alpha^i\left(\frac{W_i'}{||W_i'||}\right)U_i$$

We observe that, given the existence of K_0, k_i, then:

$$\langle K_i, K_j\rangle = k_ik_j||K_0||^2$$

$$\langle K_i', K_j'\rangle = ||K_i'|| \cdot ||K_j'|| \cdot \left\langle \frac{K_i'}{||K_i'||}, \frac{K_i'}{||K_i'||}\right\rangle$$

Since $||K_0||^2 = 1$, we know that:

$$||K_0||^2 \geq \left\langle \frac{K_i'}{||K_i'||}, \frac{K_i'}{||K_i'||}\right\rangle$$

And by the definition of K_i, we know that:

$$||K_i|| = k_i \geq ||K_i'||$$
$$||K_j|| = k_j \geq ||K_j'||$$

Therefore:

$$\langle K_i, K_j\rangle \geq \langle K_i', K_j'\rangle$$

Therefore, for any choice of vector d:

$$\left|\left|\sum_i |d_i|K_i\right|\right|^2 = \sum_{i,j}|d_id_j|\langle K_i, K_j\rangle \geq \sum_{i,j} d_id_j\langle K_i', K_j'\rangle \geq \left|\left|\sum_i d_iK_i'\right|\right|^2$$

By the definition of K_i', this implies $K_i' = K_i$. □

Lemma 5.1. *Let M be a block-diagonal matrix with blocks M_i, so that:*

$$M = \begin{pmatrix} M_1 & & \\ & ... & \\ & & M_{n_L}\end{pmatrix}$$

Let $\hat{M}$ be the M_i with the largest eigenvalue. Then $\lambda^1(M) = \lambda^1(\hat{M}), v^1(M) = v^1(\hat{M})$.

Proof. The top eigenvalue of M is found by:

$$\lambda^1(M) = \max_{||v||=1} v^TMv = \max_{||v_1||^2+...+||v_{n_L}||^2=1} \sum_i v_i^TM_iv_i$$

If we then define $v_i' = v_i/||v_i||, c_i = ||v_i||$:

$$= \max_{||c||=1}\max_{||v_1'||=...=||v_{n_L}'||=1} \sum_i c_i^2(v_i')^TM_iv_i' = \max_{||c||=1}\sum_i c_i^2\lambda^1(M_i)$$

Assume without loss of generality that M_j has the highest top eigenvalue among all the matrices M_i. The maximum is then achieved by $c_j = 1, c_i = 0$ for $i \neq j$. Therefore:

$$\lambda^1(M) = \max_i \lambda^1(M_i)$$

□

7.2 Calculation Details

7.2.1 MLPs

We use the power iteration method (34) to calculate the eigenvalues of G and Γ^i. We choose this approach over the Lanczos method (16) because it can be warm-started using the previous training iteration's eigenvectors. We set a threshold of 0.001 for terminating power iteration. We ordinarily calculate only the top eigenvalue, but since power iteration returns the highest eigenvalue by absolute value, we calculate additional eigenvalues if the returned eigenvalue is negative until we find a positive eigenvalue. When calculating a the product of a random matrix and a constant vector, we calculate the product over each sample of the random matrix, then average.

In some cases, the power iteration method failed to converge within 100 iterations when the sharpness was very small. When this occurred we registered the value as NaN, and stopped attempting to compute that value during the remainder of the experiment.

To calculate a G-vector product, we compute the network outputs z, then calculate:

$$l(z, y) - z^T \text{detach}\left(\frac{\partial l}{\partial z}\right)$$

Where detach denotes an operator that acts as the identity but prevents any gradient propagation. If we use automatic differentiation to calculate the Hessian-vector product using this pseudo-loss function, we obtain the G-vector product.

To calculate the Γ^i-vector product, we first calculate the Jacobian-vector product:

$$J_i(v) = \left(\frac{\partial \hat{x}^i}{\partial \theta}\right)^T v$$

For a vector v and for every input x^0. We then calculate:

$$g_i(v) = \frac{1}{2}\mathbb{E}\sum_j (J_i(v))_j^2 \cdot \text{detach}\left(\sigma_j''(\hat{x}^j) \cdot \frac{\partial l}{\partial x_j}\right)$$

And use autograd to calculate the gradient of $g_i(v)$. This gives us the Γ^i-vector product.

Next, for both G and Γ^i, we define the **filtered matrix** $F_{\theta^j}(G), F_{\theta^j}(\Gamma^i)$ to be the projection of the matrix to only rows and columns corresponding to θ^j - that is, to the diagonal block corresponding to layer j. Given the top eigenvectors $v^1(G), v^1(\Gamma^i)$, we calculate:

$$s_G^j = v^1(G)\big|_{\theta^j}^T F_{\theta^j}(G) v^1(G)\big|_{\theta^j}$$

Where $|$ denotes restriction of the vector to its components corresponding to a particular layer's parameters. Then:

$$\rho(G) = \frac{\lambda^1(G)}{\sum_j s_G^j}$$

We calculate $\rho(\Gamma^i)$ analogously. We then calculate the component ratios by finding the top eigenvalues of $F_{\theta^j}(G)$, and calculating:

$$\frac{\sum_j s_G^j}{\sum_j \lambda^1(F_{\theta^j)(G)}}$$

And again, we calculate the component ratios of Γ^i analogously.

7.2.2 Large Networks

Since we are calculating only a single set of results on our pre-trained networks, not tracking values over the course of training, we use the Lanczos method from hessian-eigenthings to find the eigenvalues and eigenvectors (16; 9). We use the default settings, including using the finite difference method to calculate the products instead of autograd.

We take care to replicate the loss functions and data augmentation used in training the networks:

Table 3: Deficits for Eigenvector Estimation of Large Models

Model	Deficit
VGG-16	0.006
ResNet-18	0.008

- For VGG-16 trained on ImageNet, we follow (31) in using random resize of the shortest axis to between 256 and 480 pixels while maintaining aspect ratio, followed by random crop to 224×224, random horizontal flip, and color shift following (14), using principal components of ImageNet. We modify the dropout layers (33) so that, each time a given image is passed through the network, the dropout layer drops the same entries for that image, ensuring consistency between two forward passes using the same input.
- For ResNet-18 trained on ImageNet, we follow (12) in using the same data augmentations as (31). Additionally we interpret batch normalization as a modification to the optimizer rather than as separate layers. We therefore fuse the convolutional and batch norm layers (18).

Since it is not practical to use the entire dataset when calculating the Hessian-vector product, we instead use a randomly-selected subset of 8,192 elements from each dataset. We apply the data augmentations to the subset as it is selected, so that the inputs are the same between two forward passes over the dataset. We validate that this produces a reliable estimate of the top eigenvalue and eigenvector by performing this calculation twice using different subsets, then calculating the deficit:

$$\text{deficit} = \frac{\lambda_A^1 - (v_B^1)^T G_A v_B^1}{\lambda_A^1 - \lambda_A^2}$$

Where the subscript A, B denotes which subset the term is calculated using. The deficits are shown in Table 3, and are all quite small.

We calculate the $\rho(G), \rho(\Gamma^i)$, and component ratios of the large networks analogously to how we calculated them for the MLPs, except substituting the use of the Lanczos method (16) for the power iteration method.

7.3 AI Use Disclosure

AI models were used for information retrieval and to suggest improvements to the code and paper, but the paper and code were written by humans.

7.4 Notation Glossary

$\|_{\theta^j}$	An operator which, applied to a vector whose entries correspond to the network parameters θ, returns the restriction of the vector to the entries corresponding to θ^j
α^i	The function mapping the parameters of layer i to the linear transform of layer i
$\hat{\alpha}^i$	The pushforward of α^i
detach	A function that acts as the identity on its inputs, but whose gradient is treated as identically zero
$\mathbb{E}$	Expectation of a random variable; in our case, the expectation is always taken over the network inputs randomly sampled from the training dataset $\mathcal{T}$
$F_{\theta^i}(M)$	The filtration of M to θ^i; that is, given a matrix M whose rows and columns correspond to the parameters θ, extracts the diagonal block corresponding to layer i
G	The Gauss-Newton matrix; the component of the Hessian matrix containing second derivatives of the criterion
$g_i(v)$	A function whose gradient is the product of Γ_i and v
Γ	The component of the Hessian matrix containing the second derivatives of the activation functions
Γ^i	The component of the Hessian matrix containing the second derivatives of the activation function for layer i

H	The functional Hessian; the component of the Hessian matrix containing second derivatives of the neural network
I^{α^i}	The index set of α^i, so that $I^{\alpha^i}_j$ equals those pairs (k, k') where $\alpha^i(\theta^i)_{k,k'} = \theta^i_j$
$J_i(v)$	The product of the Jacobian $\frac{\partial \hat{x}^i}{\partial \theta}$ and a vector v
$\lambda^i(M)$	The ith eigenvalue of M, where M is a symmetric matrix and so always has real eigenvalues, ordered from greatest to least
$l(z, y)$	The criterion measuring the difference between the network output z and the label y
$\widetilde{\mathcal{L}}(\theta)$	The training loss as a function of the network parameters θ
n_L	The number of network layers, considering each layer as a combined linear transform and activation function
Q^i	A random matrix equal to the product of the ith activation gradient $\frac{\partial l}{\partial x^i}$ and the Hessian of the ith activation function $\frac{\partial^2 x^i}{\partial \hat{x}^i \partial \hat{x}^i}$
R	The positive square root of the Hessian matrix of the criterion $\frac{\partial^2 l}{\partial z^2}$
$\rho(M, U)$	For a random matrices M and random vector U, the ratio of the top eigenvalue of the Gram matrix $\mathbb{E}(K')^T K'$ to the sum of its eigenvalues, where the ith column of K' is $M_i \hat{\alpha}^i(W_i'/\|\|W_i'\|\|)U_i$ and W_i' maximizes the expected squared norm of the sum of $M_i \hat{\alpha}^i(W_i)U_i$ subject to the constraint $\|\|W_1\|\|^2 + ... = 1$.
$\rho(G)$	Shorthand for $\rho\left(R\frac{\partial z}{\partial \hat{x}^i}, x^{i-1}\right)$
s^j_M	The product of $v^1(M)\|^T_{\theta^j} F_{\theta^j}(M) v^1(M)\|_{\theta^j}$
σ^i	The activation function of layer i
$\mathcal{T}$	The training dataset
θ	The parameters of the network as a whole
θ^i	The parameters of layer i
V	The component of the Hessian matrix containing the second derivatives of the products of different parameters
$v^i(M)$	The ith eigenvector of M, where M is a symmetric matrix and so always has real eigenvalues, ordered from greatest to least
x^0	The inputs to the network
x^i	The activations of layer i; the output of the activation function and input to the linear transform
$\hat{x}^i$	The preactivations of layer i; the output of the linear transform and input to the next layer's activation function
y	A label; the target the network is aiming to predict
z	A point in the space of network outputs
Z	The random matrix whose columns are Z_i
Z_0	A hypothetical random vector such that Z_i are all colinear to Z_0 for any sample of Z_0, Z_i
Z_i	The random vector $R\frac{\partial z}{\partial \hat{x}^i}\hat{\alpha}^i(v_i')x^{i-1}$, where v_i' is chosen to maximize $\mathbb{E}\|\|Z_i\|\|^2$
z_i'	A hypothetical random scalar such that $Z_i = z_i' Z_0$