On the Convex Behavior of Deep Neural Networks in Relation to the Layers' Width

Etai Littwin [1] **Lior Wolf** [1 2]

Abstract

The Hessian of neural networks can be decomposed into a sum of two matrices: (i) the positive semidefinite generalized Gauss-Newton matrix G, and (ii) the matrix H containing negative eigenvalues. We observe that for wider networks, minimizing the loss with the gradient descent optimization maneuvers through surfaces of positive curvatures at the start and end of training, and close to zero curvatures in between. In other words, it seems that during crucial parts of the training process, the Hessian in wide networks is dominated by the component G.

To explain this phenomenon, we show that when initialized using common methodologies, the gradients of over-parameterized networks are approximately orthogonal to H, such that the curvature of the loss surface is strictly positive in the direction of the gradient.

1. Introduction

It has frequently been observed (Choromanska et al., 2015) that deep neural networks are relatively easy to optimize using straightforward SGD algorithms, despite having complex non-convex loss surfaces. A great body of work in recent years has focused on the properties of SGD and its ability to maneuver through saddle points and complex surface topologies, in order to explain the relative ease of the optimization process (Daneshmand et al., 2018). A complementary approach to analyzing the optimization process, is to consider the trajectories of the gradients during optimization, with the guiding intuition that while the loss surface defined by modern neural networks might be incredibly complex and hard to analyze, the trajectories of gradient descent on these loss surfaces might prove rather simple.

An additional direction of research focuses on the spectrum of the Hessian of the loss during training, as the knowledge of the second order information about the loss surface can tell us quite a bit about the behavior of the optimizer, and how we could modify our algorithms to make them converge faster and to better solutions (Gupta et al., 2018; Martens & Grosse, 2015). Recent works have empirically shown that the distribution of the eigenvalues of the Hessian is composed of a bulk, which is concentrated around zero, and the edges which are scattered away from zero (Sagun et al., 2016). Modern neural networks, however, contain millions of parameters, resulting in a hefty Hessian with millions of eigenvalues, the overwhelming majority of which are irrelevant.

In our work, we seek to study the curvature of the Hessian in directions corresponding to the trajectory of the gradients during training, by focusing on the Hessian gradient product. Specifically, we aim to empirically and theoretically analyze the path of gradient descent algorithms by analyzing the Hessian at initialization.

We identify and empirically demonstrate novel phenomena relating to the trajectory of gradient descent neural networks under certain conditions. We observe that the gradients follow a convex path during training when performing SGD, such that the curvature of the loss is seldom negative in the direction of the gradients. This effect is more prominent in wider networks, i.e., in CNNs with a larger number of kernels. Furthermore, we theoretically prove that when using common initialization methods, the gradients of random feedforward networks are indeed pointing in a direction of positive curvature when the network is wide enough, and the input is large enough.

1.1. Terminology

Given the output of some neural network $y(x_i, w_t)$ for input x_i, with W_t denoting the model weights at iteration t, and a convex, twice differentiable loss function $\mathcal{L}(y(x_i, W_t))$, we denote the gradient on a batch of N samples by $g_t = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial \mathcal{L}(y(x_i,W_t))}{\partial W_t}$, and the Hessian by $\mathcal{H}_t = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial^2 \mathcal{L}(y(x_i,W_t))}{\partial W_t \partial W_t}$. A positive curvature during training at step t is characterized by the following condition:

$$g_t^\top \mathcal{H}_t g_t \geq 0 \tag{1}$$

[1]Tel Aviv University [2]Facebook AI Research.

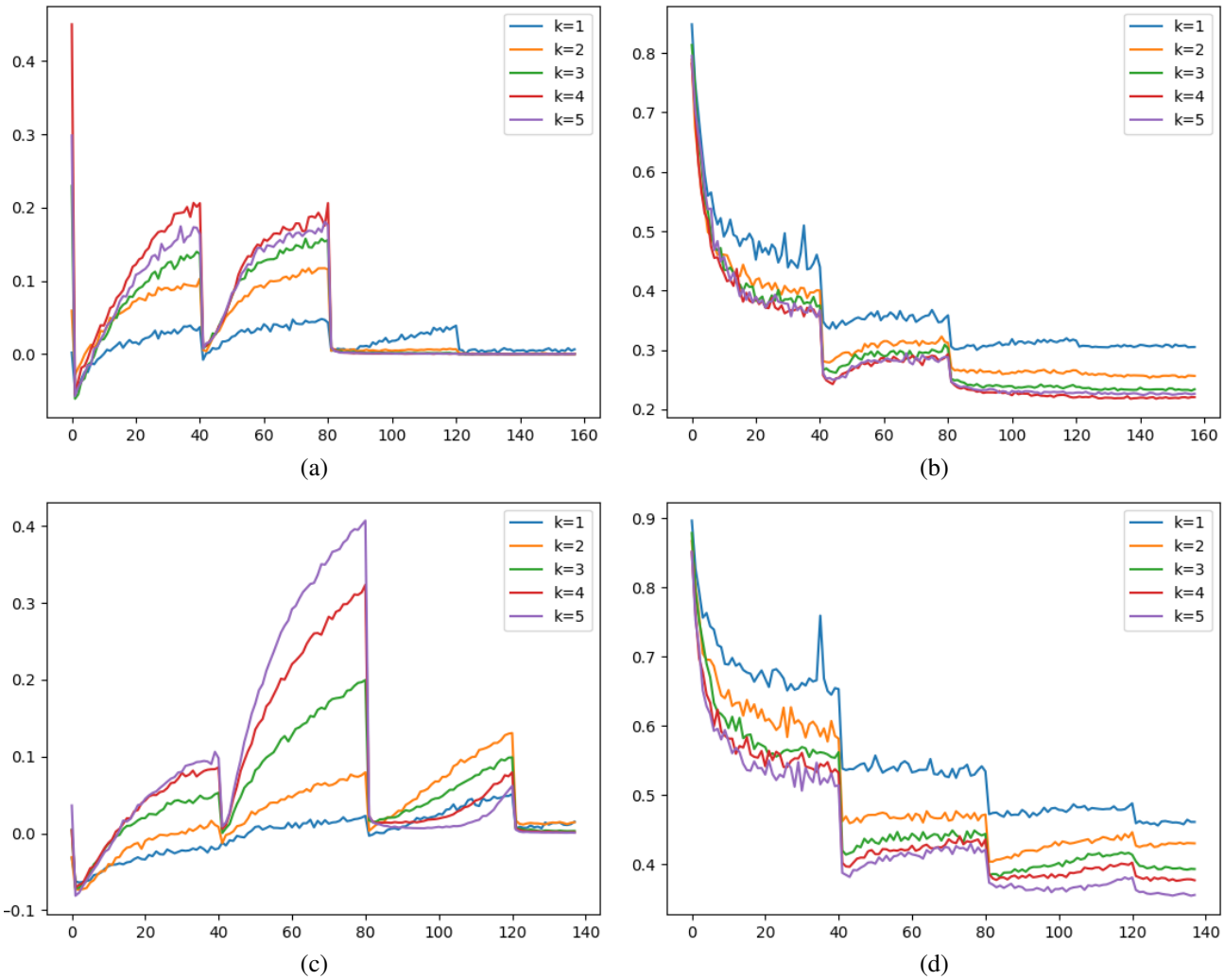

Figure 1. The approximated curvature in the direction of the gradient, for networks with different width multiplier $K = 1..5$. (a,c) as a function of the epoch for multiple network widths, shown for CIFAR-100, and tiny imagenet, respectively. The different plots denote different layer width. (b,d) the test loss for the same networks trained on CIFAR-100 and tiny imagenet.

2. Observations

We approximate the Hessian gradient product $g_t^\top \mathcal{H}_t g_t$ during the training of deep neural networks, using the following formula:

$$\mathcal{L}(W_{t+1}, \{x_i\}_{i=1}^N) \approx \mathcal{L}(W_t, \{x_i\}_{i=1}^N) - \delta \|g_t\|^2 + \delta^2 g_t^\top \mathcal{H}_t g_t \quad (2)$$

Which directly leads to:

$$g_t^\top \mathcal{H}_t g_t \approx \frac{\mathcal{L}(W_{t+1}, \{x_i\}_{i=1}^N) - \mathcal{L}(W_t, \{x_i\}_{i=1}^N)}{\delta^2} + \frac{\|g_t\|^2}{\delta} \quad (3)$$

where δ is the learning rate, and the loss $\mathcal{L}$ is computed on a minibatch $\{x_i\}_{i=1}^N$.

We track the behavior of $g_t^\top \mathcal{H}_t g_t$ across the epochs for networks of varying widths. In our experiments, we employ the ResNet-50 network (He et al., 2016), which has layers with 16, 32, or 64 kernels. We make these layers wider by multiplying the number of kernels by a factor $K = 1, \ldots, 5$. We halve the learning rate at epochs 40, 80, and 120. A batch size of 100 is used with a vanilla SGD optimizer.

The results are shown in Fig. 1 for two popular datasets: CIFAR-100 (Krizhevsky, 2009) and tiny imagenet (Le & Yang, 2015). In both datasets, one can observe that (i) throughout the majority of the training run, $g_t^\top \mathcal{H}_t g_t$ remains positive; (ii) at the start of training, and when the network converges, it obtains larger values; and (iii) typically, the wider the network, the bigger this quantity is.

Fig. 1 also presents the test error for the same benchmarks

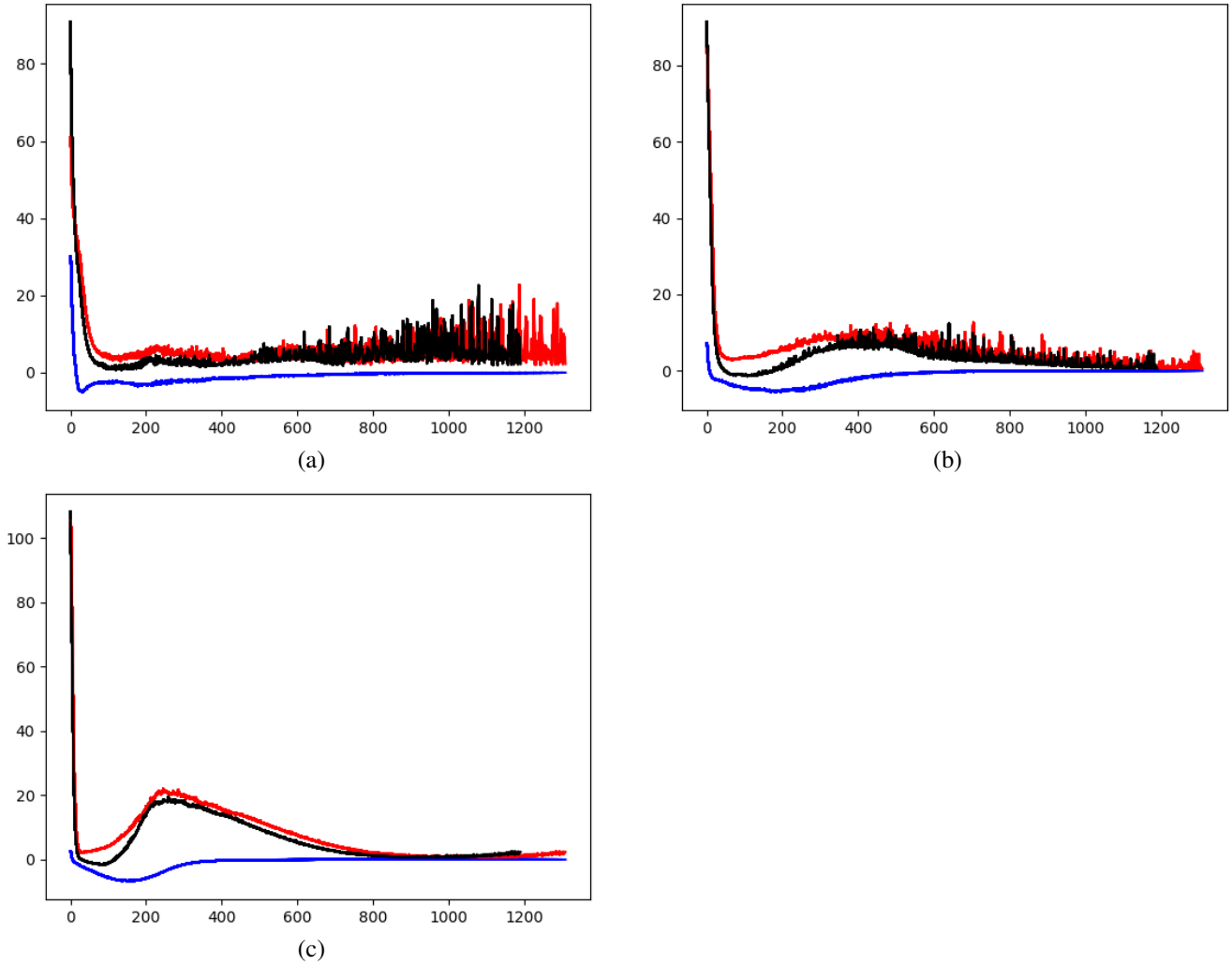

Figure 2. The curvature in the direction of the gradient, decomposed to its components $\mathcal{H}_{\hat{g}}$(black), $G_{\hat{g}}$(red) and $H_{\hat{g}}$(blue), as a function of epoch, for fully connected networks with layers' width of (a) 50, (b) 200, and (c) 400, trained on random data and labels. In all cases, $\mathcal{H}_{\hat{g}}$ is mostly dominated by $G_{\hat{g}}$, with $H_{\hat{g}}$ starting closer to zero for wider networks.

and the same networks. We observe that wider networks lead to smaller test errors, which means that their tendency toward a convex-mimicking behavior during training does not come at the expanse of the generalization ability.

We further conduct small scale experiments, in which one can efficiently decompose the empirical Hessian into its components H and G, and calculate the quantities $H_{\hat{g}} = \hat{g}^\top H \hat{g}$, $G_{\hat{g}} = \hat{g}^\top G \hat{g}$ and $\mathcal{H}_{\hat{g}} = \hat{g}^\top \mathcal{H} \hat{g}$ during training. Specifically, we train a single output, 8 layer fully connected network with layer width of 50, 200 and 400 on a set of 1000 predetermined random inputs sampled from a Gaussian distribution with dimensionality matching the width of the network, which are then normalized to have unit norm. The network are trained to fit with a standard L2 loss random labels sampled from a binary distribution.

The results are shown in Fig. 2. As can be seen, in all cases, the network exhibits a convex-mimicking behaviour, similar to the results of our CNN experiments. Specifically, the curvature observed in $\mathcal{H}_{\hat{g}}$ seems to closely follow that of $G_{\hat{g}}$. In addition, as predicted in our analysis, the curve of $H_{\hat{g}}$ starts closer to zero for wider networks.

3. Analysis

An initial analysis is performed for linear networks (no non-linearities) at initialization, where the weights of the networks are randomly sampled. We make very few assumptions on the nature of the networks' initialization, and the popular initialization techniques satisfy our assumptions.

3.1. Overview

In our theoretical analysis, we focus on the Hessian of a single output linear network with L layers, where the output of layer $l = 1, \dots, L$ is given by $y^l = W^{l\top} y^{l-1}$ where $W^l \in \mathbb{R}^{n_{l-1} \times n_l}, n_L = 1, y^l \in \mathbb{R}^{n_l}$, and the input, which is also denoted by y^0 is given by a fixed dataset $\{x_s\}_{s=1}^N, x_s \in \mathbb{R}^{n_0}$. We denote by W the concatenation of all the parameters of the network in a column vector.

As mentioned, we consider a twice differentiable convex loss function $\mathcal{L}$. We further assume that the loss has bounded first and second derivatives. The Hessian matrix is given by:

$$\begin{aligned}\mathcal{H} &= \sum_{s=1}^{N} \mathcal{L}''(y(x_s)^L) g_w^s g_w^{s\top} + \sum_{s=1}^{N} \mathcal{L}'(y(x_s)^L) \frac{\partial^2 y^L(x_s)}{\partial W \partial W} \\ &= \sum_{s=1}^{N} \mathcal{L}''(y(x_s)^L) \hat{G}^s + \sum_{s=1}^{N} \mathcal{L}'(y(x_s)^L) \hat{H}^s = G + H\end{aligned} \tag{4}$$

where g_w^s is the gradient of the network output unit for input s with respect to the weights. The first term of the Hessian is comprised of a sum of positive semidefinite matrices, and is, therefore, positive semidefinite, while the second term is not, and is dependent on the architecture. In the following, we aim to show that under certain conditions pretaining to the width of the network and the dimensionality of the input, which are usually met in practice, the trajectory of GD is influenced primarily by G. That is, denoting by index t the evaluation of a tensor at iteration t, it holds that:

$$g_t^\top \mathcal{H}_t g_t \approx g_t^\top G_t g_t > 0 \tag{5}$$

3.2. Main Results

In our analysis, we consider the weights $W^k = [w_{ij}^k]_{0<i\le n_k, 0<j\le n_{k-1}, 0<k\le L}$ sampled iid from a symmetric distribution with moments m_t, such that $m_2, m_4, m_6 \approx \frac{1}{n_{k-1}}$, where $m_t = \mathbb{E}\left((w_{ij}^k)^t\right)$ are the moments of the distribution. Notice that this condition applies to the common initialization scheme (He et al., 2015). We first consider the gradient and Hessian evaluated on a single sample with unit norm, i.e., $\|y^0\| = 1$, and then discuss multiple samples. We make the following notations: the derivative of the output unit with respect to a single weight ij in layer k is denoted by $g_{ij}^k = \frac{\partial y^L}{\partial w_{ij}^k} = \frac{\partial y^L}{\partial y^k} \frac{\partial y^k}{\partial w_{ij}^k}$.

we denote $\hat{H}$ by:

$$\hat{H}_{ijk}^{uvl} = \frac{\partial^2 y^L}{\partial w_{ij}^k \partial w_{uv}^l} \tag{6}$$

where the triplets of indices ijk, uvl parameterize row and column indices in the matrix $\hat{H}$ (note that the ranges of ij, uv depend on the layer indices k and l). The next theorem gives the expected mean and variance of $g^\top \hat{H} g$ for wide networks at initialization:

Theorem 1. $\mathbb{E}\left(g^T \hat{H} g\right) = 0$ *and for* $min_{l<L}(n_l) >> L$, *it holds that* $\mathbb{E}\left((g^T \hat{H} g)^2\right) \approx \Theta(\frac{\sum_{k=1}^L \sum_{k'=1}^L n_k n_{k'}}{n_0^3})$

The Hessian gradient product is not necessarily indicative of the local curvature, since we do not take into account the norm of the gradient. As shown in (Arpit & Bengio, 2019), a constant width network ensures that the norm of the gradient approximately equals the input. More specifically, by having $\forall_{0\le l\le L},\ n_l = n m_l$, we ensure that $\|g\|^2 \to \frac{\sum_{l=1}^L m_l}{m_0} \|y^0\|^2$ as n gets larger. (The precise statement is omitted for brevity). The next theorem gives a bound on the probability of the local curvature to be above some ϵ, given the full Hessian.

Theorem 2. *Assume that* $\forall_{0\le l\le L},\ n_l = n m_l$, *and* $Prob(\frac{\sum_{l=1}^L m_l}{m_0} \|y^0\|^2 - \epsilon \le \|g\|^2 \le \frac{\sum_{l=1}^L m_l}{m_0} \|y^0\|^2 + \epsilon) > 1 - \delta(\epsilon)$, $\mathcal{L}''(y^L) > \alpha > 0, \mathcal{L}'(y^L) > \beta > 0$ *then it holds that:*

$$\begin{aligned}Prob\left(\hat{g}^\top \mathcal{H} \hat{g} > \epsilon\right) &\ge (1 - \delta(\frac{2\epsilon}{\|\alpha\|}))(1 - \delta(\epsilon)) \\ &(1 - \frac{1}{n} \frac{\gamma \sum_{kk'=1}^L m_k m_{k'}}{m_0^3 (\frac{\epsilon}{\|\beta\|})^2 (\frac{\sum_{l=1}^L m_l}{m_0} \|y^0\|^2 - (\frac{\epsilon}{\|\beta\|}))^2})\end{aligned} \tag{7}$$

The above analysis provides a rather surprising result, which is that the dimensionality of the input also plays a role in determining the local curvature at initialization, in addition to the width of the network.

We have discussed a single sample in our analysis so far. In the case of linear neural networks, the result of Thm. 1 also hold for the batch case since we did not assume anything on the input. Specifically, since $H = \sum_{s=1}^N \mathcal{L}'(y(x_s)^L) \hat{H}^s$, and considering the gradient of sample u, and $\hat{H}^v$ evaluated on sample v, then it holds that $g^{u\top} \hat{H}^v g^u \approx \frac{\sum_{kk'} n_k n_{k'}}{n_0^3}$.

The term $g^\top G g$ can be significantly smaller in the batch case, even when the gradient is large. Indeed, under mild conditions, when gradients of individual samples are orthogonal to each other, this term can be made small while keeping the overall gradient large, insuring a faster convergence to the optimum. Precise statement and analysis are kept for future work.

4. Discussion

The literature has already linked the width of a neural network to the property of norm-preserving. For example, (Arpit & Bengio, 2019) show that under different sets of assumptions, the wider the network, the more likely the

norm of the activations will be preserved with depth. Our work is the first one, as far as we know, to show a link between a convex-mimicking behavior during optimization and the network's width.

The literature also teaches us about the properties of the Hessian during training (Sagun et al., 2016). As far as we know, we are unique in that we look at the Hessian in the direction of the gradient, where it matters most to the optimization process.

The line of research can lead to a better understanding of the link between different architectures and their trainability. Moreover, the signal we track can be readily used in order to drive the optimization process. Specifically, it may be derivable to adjust the hyperparameters (learning rate, batch size, batch composition, etc.) such that the gradients are large while the product of the gradient with the Hessian implies close to zero curvature.

Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant ERC CoG 725974).

References

Arpit, D. and Bengio, Y. The benefits of over-parameterization at initialization in deep relu networks. *arXiv preprint arXiv:1901.03611*, 2019.

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. The Loss Surfaces of Multilayer Networks. In Lebanon, G. and Vishwanathan, S. V. N. (eds.), *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pp. 192–204, 2015.

Daneshmand, H., Kohler, J., Lucchi, A., and Hofmann, T. Escaping saddles with stochastic gradients. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1155–1164, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/daneshmand18a.html.

Gupta, V., Koren, T., and Singer, Y. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pp. 1837–1845, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pp. 1026–1034, Washington, DC, USA, 2015. IEEE Computer Society. ISBN 978-1-4673-8391-2. doi: 10.1109/ICCV.2015.123. URL http://dx.doi.org/10.1109/ICCV.2015.123.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Le, Y. and Yang, X. J. Tiny imagenet visual recognition challenge. Technical report, 2015.

Martens, J. and Grosse, R. B. Optimizing neural networks with kronecker-factored approximate curvature. In *ICML*, 2015.

Sagun, L., Bottou, L., and LeCun, Y. Singularity of the hessian in deep learning. *arXiv preprint arXiv:1611.07476*, 2016.

A. Proof of Thm. 1

Proof. The derivative of unit j in layer k with respect to unit u in layer l is denoted by $\frac{\partial y_j^k}{\partial y_u^l} = a_{lu}^{kj}$. Dropping the unit indexes j or u means deriving the entire layer, or with respect to the entire layer respectively. Dropping all superscripts means derivative of the output unit, so that $\frac{\partial y_L}{\partial y_u^l} = a_{lu}$. It then holds that $g_{ij}^k = a_{ki} y_j^{k-1}$. Finally, we denote by $W_{,i}^l$ and $W_{i,}^l$ the $i'th$ column and row vectors of W^l respectively. We have:

$$\hat{H}_{ijk}^{uvl} = \frac{\partial^2 y^L}{\partial w_{ij}^k \partial w_{uv}^l} = \{ \begin{array}{cc} a_{ki} a_{lu}^{k-1j} y_v^{l-1} & l < k-1 \\ a_{ki} y_v^{l-1} \delta_{ju} & l = k-1 \\ 0 & l = k \\ a_{lu} y_j^{k-1} \delta_{vi} & l = k+1 \\ a_{lu} a_{ki}^{l-1v} y_j^{k-1} & l > k+1 \end{array} \tag{8}$$

where the triplets of indices ijk, uvl parameterize row and column indices in the matrix $\hat{H}$ (note that the ranges of ij, uv depend on the layer indices k and l).

The Hessian gradient product is given by:

$$\sum_{ijk} \hat{H}_{ijk}^{uvl} g_{ij}^k = \{ \begin{array}{cc} \sum_i (a_{2i})^2 y_v^0 y_j^1 \delta_{ju} + \sum_{ijk>2} (a_{ki})^2 a_{lu}^{k-1j} y_v^0 y_j^{k-1} & l = 1 \\ \sum_j a_{2u} a_{1i} (y_j^0)^2 \delta_{vi} + \sum_i (a_{3i})^2 y_v^1 y_j^2 \delta_{ju} + \sum_{ijk>3} (a_{ki})^2 a_{2u}^{k-1j} y_v^1 y_j^{k-1} & l = 2 \\ \begin{array}{c} \sum_{ijk>l+1} (a_{ki})^2 a_{lu}^{k-1j} y_v^{l-1} y_j^{k-1} + \sum_{ijk<l-1} a_{lu} a_{ki}^{l-1v} a_{ki} (y_j^{k-1})^2 \\ + \sum_i (a_{l+1i})^2 y_v^{l-1} y_j^l \delta_{ju} + \sum_j a_{lu} a_{l-1i} (y_j^{l-2})^2 \delta_{vi} \end{array} & L-1 > l > 2 \\ \begin{array}{c} \sum_j (a_{Lj})^2 y_v^{L-2} y_j^{L-1} \delta_{ju} + \sum_{ij} a_{L-1u} a_{L-2i} (y_j^{L-3})^2 \delta_{vi} \\ + \sum_{ijk<L-2} a_{L-1u} a_{ki} a_{ki}^{L-2v} (y_j^{k-1})^2 \end{array} & l = L-1 \\ \sum_{ij} a_{Lu} a_{L-1i} (y_j^{L-2})^2 \delta_{vi} + \sum_{ijk<L-1} a_{Lu} a_{ki} a_{ki}^{L-1v} (y_j^{k-1})^2 & l = L \end{array} \tag{9}$$

The claim $\mathbb{E}\left(g^T \hat{H} g\right) = 0$ is trivially true, since each term in the Hessian gradient products will contain odd moments of the weights, which zero out for symmetric distributions. For deep networks, we have that:

$$\mathbb{E}\left((g^\top \hat{H} g)^2\right) = \mathbb{E}\left((\sum_{uvl} \sum_{ijk} g_{uv}^l \hat{H}_{ijk}^{uvl} g_{ij}^k)^2 \right) \approx \mathbb{E}\left((\sum_{uv, L-1>l>2} \sum_{ij, k \neq [l-1, l+1]} g_{uv}^l \hat{H}_{ijk}^{uvl} g_{ij}^k)^2 \right) \tag{10}$$

and so:

$$\sum_{\substack{uv, \\ L-1>l>2}} \sum_{\substack{ij, \\ k \neq [l-1, l+1]}} g_{uv}^l \hat{H}_{ijk}^{uvl} g_{ij}^k = \sum_{uvijk>l+1} (a_{ki})^2 a_{lu} a_{lu}^{k-1j} (y_v^{l-1})^2 y_j^{k-1} + \sum_{uvijk<l-1} (a_{lu})^2 a_{ki}^{l-1v} a_{ki} (y_j^{k-1})^2 y_v^{l-1} \tag{11}$$

$$\begin{aligned} \mathbb{E}\left((g^\top \hat{H} g)^2\right) &\approx \mathbb{E}\left((\sum_{uvijk>l+1} (a_{ki})^2 a_u^l a_{lu}^{k-1j} (y_v^{l-1})^2 y_j^{k-1} + \sum_{uvijk<l-1} (a_{lu})^2 a_{ki}^{l-1v} a_{ki} (y_j^{k-1})^2 y_v^{l-1})^2 \right) \\ &\approx \mathbb{E}\left((\sum_{uvijk>l+1} \sum_{u'v'i'j'k'>l'+1} (a_{ki})^2 (a_{k'i'})^2 a_{lu} a_{l'u'} a_{lu}^{k-1j} a_{l'u'}^{k'-1j'} (y_v^{l-1})^2 (y_{v'}^{l'-1})^2 y_j^{k-1} y_{j'}^{k'-1} \right) \\ &\approx \mathbb{E}\left(\sum_{uviju'v'i'j} (a_{ki})^2 (a_{k'i'})^2 a_{lu} a_{l'u'} a_{lu}^{k-1j} a_{l'u'}^{k'-1j'} (y_v^{l-1})^2 (y_{v'}^{l'-1})^2 y_j^{k-1} y_{j'}^{k'-1} \right) \end{aligned} \tag{12}$$

where we assumed in the last transition that $min_{l<L}(n_l) >> L$. Applying expectation on the last weight layer W^L, it holds

that:

$$\mathbb{E}\left(\sum_{uvijv'i'j}(a_{ki})^2(a_{k'i'})^2a_{lu}a_{l'u'}\right)$$
$$\approx\frac{1}{n_{L-1}^3}\mathbb{E}\Bigg(\sum_{uvijv'i'j}(\|a_{ki}^{L-1}\|^2\|a_{k'i'}^{L-1}\|^2a_{lu}^{L-1\top}a_{l'u'}^{L-1}+(a_{ki}^{L-1\top}a_{k'i'}^{L-1})^2a_{lu}^{L-1\top}a_{l'u'}^{L'-1}$$
$$+(a_{ki}^{L-1\top}a_{k'i'}^{L-1})a_{k'i'}^{L'-1\top}a_{lu}^{L-1}a_{ki}^{L-1\top}a_{l'u'}^{L'-1}+(a_{ki}^{L-1\top}a_{k'i'}^{L-1})a_{k'i'}^{L'-1\top}a_{l'u'}^{L'-1}a_{ki}^{L-1\top}a_{lu}^{L-1})\Bigg) \quad (13)$$

Plugging each term in Eq. 12 yields the same asymptotic behaviour. For the sake of brevity, we will show this with the first term. Without loss of generality, we assume $k>k'+1>l+2>l'+3$, we now recursively apply expectations over $W^{L-1}...W^k$, while noticing that the following hold for fixed vectors $v_1...v_6$:

$$\mathbb{E}\Big(v_1^\top W^{l\top}W^lv_2v_3^\top W^{l\top}W^lv_4v_5^\top W^{l\top}W^lv_6\Big)\approx\frac{n_l^3}{n_{l-1}^3}v_1^\top v_2v_3^\top v_4v_5^\top v_6$$
$$\mathbb{E}\Big(v_1^\top W_{j,}^{l\top}W_{,j}^lv_2v_3^\top W^{l\top}W^lv_4v_5^\top W^{l\top}W^lv_6\Big)\approx\frac{n_l^2}{n_{l-1}^3}v_1^\top v_2v_3^\top v_4v_5^\top v_6$$
$$\mathbb{E}\Big(W_{,j}^{l\top}W^lv_1W_{,j}^{l\top}W^lv_2v_3^\top W^{l\top}W^lv_4\Big)\approx\frac{n_l^2}{n_{l-1}^3}v_1^\top v_2v_3^\top v_4 \quad (14)$$

and so:

$$\mathbb{E}\left((g^\top\hat{H}g)^2\right)\approx\frac{1}{n_k^3}\mathbb{E}\Bigg(\sum_{uvijv'i'j}(\|a_{ki}^k\|^2\|a_{k'i'}^k\|^2a_{lu}^{k\top}a_{l'u'}^ka_{lu}^{k-1j}a_{l'u'}^{k'-1j'}(y_v^{l-1})^2(y_{v'}^{l'-1})^2y_j^{k-1}y_{j'}^{k'-1})\Bigg)$$
$$\approx\frac{1}{n_{k-1}^3}\mathbb{E}\Bigg(\sum_{uvijv'i'j}(\|a_{k'i'}^{k-1}\|^2a_{lu}^{k-1\top}a_{l'u'}^{k-1}a_{lu}^{k-1j}y_j^{k-1}a_{l'u'}^{k'-1j'}(y_v^{l-1})^2(y_{v'}^{l'-1})^2y_{j'}^{k'-1})\Bigg)$$
$$\approx\frac{1}{n_{k-1}n_{k-2}^3}\mathbb{E}\Bigg(\sum_{uvijv'i'j}(\|a_{k'i'}^{k-2}\|^2a_{lu}^{k-2\top}a_{l'u'}^{k-2}a_{lu}^{k-2\top}y^{k-2}a_{l'u'}^{k'-1j'}(y_v^{l-1})^2(y_{v'}^{l'-1})^2y_{j'}^{k'-1})\Bigg) \quad (15)$$

Recursively going through $l=k-2...1$ we have:

$$\mathbb{E}\left((g^\top\hat{H}g)^2\right)\approx\frac{\|y^0\|^6}{n_{k-1}n_{k'-1}n_{l-1}n_{l'-1}n_ln_{l'}n_0^3}\sum_{uvijv'i'j}=\frac{n_{k-1}n_kn_{k'-1}n_k'n_{l-1}n_ln_{l'-1}n_l'}{n_{k-1}n_{k'-1}n_{l-1}n_{l'-1}n_ln_{l'}n_0^3}=\frac{n_kn_{k'}}{n_0^3} \quad (16)$$

Summing over k,k':

$$\mathbb{E}\left((g^\top\hat{H}g)^2\right)\approx\frac{\sum_{kk'}n_kn_{k'}}{n_0^3} \quad (17)$$

□

B. Proof of Thm. 2

Proof. We have:

$$Prob\Big(\hat{g}^\top\mathcal{H}\hat{g}>\epsilon\Big)\geq Prob\Big(\hat{g}^\top G\hat{g}>2\epsilon\Big)Prob\Big(|\hat{g}^\top H\hat{g}|<\epsilon\Big)$$
$$=Prob\Big(\mathcal{L}''\|g\|^2>2\epsilon\Big)Prob\Big(|\mathcal{L}'||\hat{g}^\top\hat{H}\hat{g}|<\epsilon\Big)\geq Prob\Big(|\alpha|\|g\|^2>2\epsilon\Big)Prob\Big(|\beta||\hat{g}^\top\hat{H}\hat{g}|<\epsilon\Big) \quad (18)$$

$$Prob\Big(|\hat{g}^\top \hat{H}\hat{g}| < \epsilon\Big)$$
$$\geq Prob\Big(\frac{1}{\frac{\sum_{l=1}^{L} m_l}{m_0}\|y^0\|^2 + \epsilon} \leq \frac{1}{\|g\|^2} \leq \frac{1}{\frac{\sum_{l=1}^{L} m_l}{m_0}\|y^0\|^2 - \epsilon}\Big) Prob\Big(|g^\top \hat{H} g| < \epsilon(\frac{\sum_{l=1}^{L} m_l}{m_0}\|y^0\|^2 - \epsilon)\Big) \quad (19)$$

From The. 1, we have that $var(g^\top \hat{H} g) = \frac{\gamma \sum_{kk'=1}^{L} m_k m_{k'}}{n m_0^3}$ for some $\gamma > 0$. Using Chebyshev's inequality:

$$Prob\Big(|g^\top \hat{H} g| < \epsilon(\frac{\sum_{l=1}^{L} m_l}{m_0}\|y^0\|^2 - \epsilon)\Big) > 1 - \frac{1}{n}\frac{\gamma \sum_{kk'=1}^{L} m_k m_{k'}}{m_0^3 \epsilon^2 (\frac{\sum_{l=1}^{L} m_l}{m_0}\|y^0\|^2 - \epsilon)^2} \quad (20)$$

and so:

$$Prob\Big(|\hat{g}^\top \hat{H}\hat{g}| < \epsilon\Big) \geq (1 - \delta(\epsilon))(1 - \frac{1}{n}\frac{\gamma \sum_{kk'=1}^{L} m_k m_{k'}}{m_0^3 \epsilon^2 (\frac{\sum_{l=1}^{L} m_l}{m_0}\|y^0\|^2 - \epsilon)^2}) \quad (21)$$

Finally:

$$Prob\Big(\hat{g}^\top \mathcal{H}\hat{g} > \epsilon\Big) \geq (1 - \delta(\frac{2\epsilon}{\|\alpha\|}))(1 - \delta(\epsilon))(1 - \frac{1}{n}\frac{\gamma \sum_{kk'=1}^{L} m_k m_{k'}}{m_0^3 (\frac{\epsilon}{\|\beta\|})^2 (\frac{\sum_{l=1}^{L} m_l}{m_0}\|y^0\|^2 - (\frac{\epsilon}{\|\beta\|}))^2}) \quad (22)$$

□