A New Perspective on Shampoo's Preconditioner

Depen Morwani*
SEAS
Harvard University
dmorwani@g.harvard.edu

Itai Shapira*
SEAS
Harvard University
itaishapira@g.harvard.edu

Nikhil Vyas*
SEAS
Harvard University
nikhil@g.harvard.edu

Eran Malach
Kempner Institute
Harvard University
emalach@g.harvard.edu

Sham Kakade
Kempner Institute
Harvard University
sham@seas.harvard.edu

Lucas Janson
Department of Statistics
Harvard University
ljanson@g.harvard.edu

Abstract

Shampoo, a second-order optimization algorithm which uses a Kronecker product preconditioner, has recently garnered increasing attention from the machine learning community. The preconditioner used by Shampoo can be viewed either as an approximation of the Gauss–Newton component of the Hessian or the covariance matrix of the gradients maintained by Adagrad. We provide an explicit and novel connection between the *optimal* Kronecker product approximation of these matrices and the approximation made by Shampoo. Our connection highlights a subtle but common misconception about Shampoo's approximation. In particular, the *square* of the approximation used by the Shampoo optimizer is equivalent to a single step of the power iteration algorithm for computing the aforementioned optimal Kronecker product approximation. Across a variety of datasets and architectures we empirically demonstrate that this is close to the optimal Kronecker product approximation. Additionally, for the Hessian approximation viewpoint, we empirically study the impact of various practical tricks to make Shampoo more computationally efficient (such as using the batch gradient and the empirical Fisher) on the quality of Hessian approximation.

1 Introduction

Second-order optimization is a rich research area within deep learning that has seen multiple influential works over the past few decades. Recently, these methods have seen success in practical large scale training runs such as Gemini 1.5 Flash (Gemini Team, 20024) and in academic benchmarks (Dahl et al., 2023). One of the primary challenges in this field arises from the substantial memory and computational demands of traditional second-order methods, such as Adagrad (Duchi et al., 2011b) and Newton's method. In the context of neural networks, both of these methods require storing and inverting a $|P| \times |P|$ dimensional matrix H (either covariance of the gradients for Adagrad or the Gauss–Newton component of the Hessian for Newton's method), where $|P|$ represents the number of parameters of the neural network. With modern deep learning architecture scaling to billions of parameters, these requirements make the direct application of these methods impractical. To address this issue, various approaches have been proposed, including Hessian-free optimization (Martens et al., 2010) and efficient approximations of the matrix H (Gupta et al., 2018b; Martens & Grosse, 2015b). These methods aim to leverage second-order information while mitigating the computational and memory overhead.

*Equal contribution. Randomized Author Ordering.

Preprint. Under review.

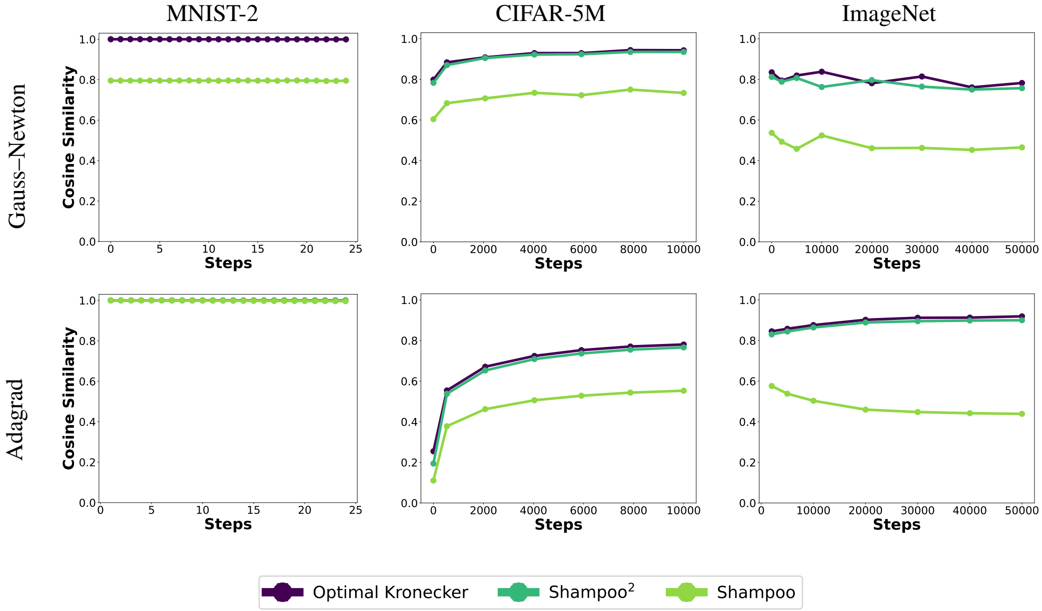

Figure 1: Top: Cosine similarity between different approximations of the Gauss–Newton (GN) component of the Hessian and its true value for different datasets and architectures. Bottom: Similar plot showing the cosine similarity between different approximations of the Adagrad preconditioner matrix and its true value. As can be seen, Shampoo^2 tracks the optimal Kronecker approximation much more closely than Shampoo does. MNIST-2 refers to a binary subsampled MNIST dataset. For more details about datasets and architectures, please refer to Appendix B.

The class of methods for efficiently approximating the matrix H predominantly involve either a diagonal or a layer-wise Kronecker product approximation of H. These choices are motivated by the fact that, compared to maintaining the matrix H, both diagonal and layer-wise Kronecker products are significantly more memory-efficient to store and computationally efficient to invert. Two of the most well-known methods that utilize a layer-wise Kronecker product approximation of H are K-FAC (Martens & Grosse, 2015b) and Shampoo (Gupta et al., 2018b).

In this work, we primarily focus on the Shampoo optimizer (Gupta et al., 2018b), which has recently gained increasing attention from the research community. Notably, in a recent benchmark of optimization algorithms proposed for practical neural network training workloads (Dahl et al., 2023), Shampoo appears to outperform all other existing methods. Another recent study, elucidating the Google Ads recommendation search pipeline, revealed that the Google Ads CTR model is trained using the Shampoo optimizer (Anil et al., 2022). Additionally, a recent work (Shi et al., 2023) implemented a distributed data parallel version of Shampoo, demonstrating its superior speed in training ImageNet compared to other methods.

Previously, Shampoo's approximation was shown to be an upper bound (in spectral norm) on the matrix H (Gupta et al., 2018b). In this work, we make this connection much more precise. Prior research has established the notion of the optimal Kronecker product approximation (in Frobenius norm) of H (Koroko et al., 2023b), which can be obtained numerically using a power iteration scheme. The primary contribution of this work is to theoretically and empirically demonstrate that the square of the approximation used by Shampoo is nearly equivalent to the optimal Kronecker factored approximation of H.

The main contributions of the work are summarized below:

- We theoretically show (Proposition 1) that the square of the Shampoo's approximation of H is precisely equal to one round of the power iteration scheme for obtaining the optimal Kronecker factored approximation of the matrix H. Informally, for any covariance matrix

$H = \mathbb{E}[gg^T]$ where $g \in \mathbb{R}^{mn}$ [2], we argue that the *right* Kronecker product approximation of H is $\mathbb{E}[GG^\top] \otimes \mathbb{E}[G^\top G]$ while Shampoo proposes $\mathbb{E}[GG^\top]^{1/2} \otimes \mathbb{E}[G^\top G]^{1/2}$, with $G \in \mathbb{R}^{m \times n}$ representing a reshaped g into a matrix of size $m \times n$.

- We empirically establish that the result of one round of power iteration is very close to the optimal Kronecker factored approximation (see Figure 1), and provide theoretical justification for the same.
- For the Hessian based viewpoint of Shampoo (Section 2.1.2), we empirically demonstrate the impact on the Hessian approximation of various practical tricks implemented to make Shampoo more computationally efficient such as averaging gradients over batch (Section 4.1) and using empirical Fisher instead of the actual Fisher (Section 4.2).

Remark. Previous works (Balles et al., 2020; Lin et al., 2024) have explored the question of why Adagrad-based approaches like Adam and Shampoo have an extra square root compared to the Hessian inverse in their update. This alternative question is orthogonal to our contribution. For details, refer Appendix F.

Paper organization. In Section 2, we cover the technical background necessary for understanding this work. In Section 3, we provide a general power iteration scheme for obtaining the optimal Kronecker product approximation of the matrix H, and establish the the connection between Shampoo's approximation and the optimal Kronecker product approximation of H. In Section 4, we explore the Hessian approximation viewpoint of Shampoo and empirically study how various practical tricks to make Shampoo more computationally efficient impact the quality of the Hessian approximation. In Section 5, we cover closely related works and conclude with discussing the limitations of the work in Section 6. In Appendix A, we include additional experiments on the ViT architecture and compare with the K-FAC approximation to the Hessian. Detailed related work, proofs, dataset and architecture details have been deferred to the Appendix.

2 Technical background

We use lowercase letters to denote scalars and vectors, and uppercase letters to denote matrices. For a symmetric matrix A, $A \succcurlyeq 0$ (resp. $A \succ 0$) denotes that A is positive semi-definite (resp. positive definite). Similarly, for symmetric matrices A and B, $A \succcurlyeq B$ (resp. $A \succ B$) denotes $A - B \succcurlyeq 0$ (resp. $A - B \succ 0$). We will use $M[i, j]$ refer to the 0-indexed (i, j) entry of the matrix M. The Kronecker product of two matrices $A \in \mathbb{R}^{p \times q}$ and $B \in \mathbb{R}^{r \times s}$ is denoted by $A \otimes B \in \mathbb{R}^{pr \times qs}$. It is defined such that $(A \otimes B)[ri + i', sj + j'] = A[i, j]B[i', j']$ where $0 \leqslant i < p, 0 \leqslant j < q, 0 \leqslant i' < r, 0 \leqslant j' < s$. Vectorization of a matrix $A \in \mathbb{R}^{m \times n}$, denoted by $\mathrm{vec}(A)$, is a mn-dimensional column vector obtained by stacking the columns of A on top of one another. We will usually denote $\mathrm{vec}(A)$ by a.

Following is a basic lemma about Kronecker products that will be used later

Lemma 1 (Henderson & Searle (1981))**.** $(A \otimes B) \,\mathrm{vec}(G) = \mathrm{vec}(BGA^\top)$.

2.1 Shampoo

The original Shampoo (Gupta et al., 2018b) paper introduced its algorithm as an approximation of an online learning algorithm Adagrad (Duchi et al., 2011a). Shampoo can also be interpreted (Anil et al., 2020; Osawa et al., 2023a) as approximating the Gauss–Newton component of the Hessian. Both of these perspectives will be discussed in Section 2.1.1 and 2.1.2 respectively. .

2.1.1 Adagrad based perspective of Shampoo

Adagrad: This is a preconditioned online learning algorithm, that uses the accumulated covariance of the gradients as a preconditioner. Let $\theta_t \in \mathbb{R}^p$ denote the parameters at time t and let $g_t \in \mathbb{R}^p$ denote the gradient. It maintains a preconditioner $H_{\text{Ada}} = \sum_{t=1}^{T} g_t g_t^\top$. The update for the parameter for learning rate η are given by

[2] Gauss–Newton component of the Hessian can also be expressed as a covariance matrix. For details, refer Section 2.1.2

$$\theta_{T+1} = \theta_T - \eta H_{\text{Ada}}^{-1/2} g_T.$$

Shampoo is a preconditioned gradient method which maintains a layer-wise Kronecker product approximation to full-matrix Adagrad. Let the gradient for a weight matrix[3] $W_t \in \mathbb{R}^{m \times n}$ at time t be given by $G_t \in \mathbb{R}^{m \times n}$. The lemma below is used to obtain the Shampoo algorithm from Adagrad:

Lemma 2 (Gupta et al. (2018b))**.** *Assume that $G_1, ..., G_T$ are matrices of rank at most r. Let $g_t = \text{vec}(G_t)$ for all t. Then, with $\preccurlyeq$ representing the for any $\epsilon > 0$,*

$$\epsilon I_{mn} + \frac{1}{r}\sum_{t=1}^{T} g_t g_t^\top \preccurlyeq \left(\epsilon I_m + \sum_{t=1}^{T} G_t G_t^\top\right)^{1/2} \otimes \left(\epsilon I_n + \sum_{t=1}^{T} G_t^\top G_t\right)^{1/2}.$$

Based on the above lemma, Shampoo maintains two preconditioners $L_t \in \mathbb{R}^{m \times m}$ and $R_t \in \mathbb{R}^{n \times n}$, which are initialized to ϵI_m and ϵI_n respectively. . The update for the preconditioners and the Shampoo update for a learning rate η is given by

$$L_T = L_{T-1} + G_T G_T^\top; \quad R_T = R_{T-1} + G_T^\top G_T; \quad W_{T+1} = W_T - \eta L_T^{-1/4} G_T R_T^{-1/4}.$$

In Lemma 2 the matrix $H_{\text{Ada}} = \sum_{t=1}^{T} g_t g_t^\top$ is approximated (ignoring ϵ and scalar factors) by the the Kronecker product $\left(\sum_{t=1}^{T} G_t G_t^\top\right)^{1/2} \otimes \left(\sum_{t=1}^{T} G_t^\top G_t\right)^{1/2}$. Our main focus will be to study the *optimal Kronecker product approximation* of the matrix H_{Ada} and its connection to Shampoo's approximation (done in Section 3).

2.1.2 Hessian based perspective of Shampoo

In this section we describe the Hessian approximation viewpoint of Shampoo explored by previous works (Anil et al., 2020; Osawa et al., 2023a) as an alternative to the Adagrad viewpoint described above. Our theoretical and empirical results hold for both viewpoints.

Gauss–Newton (GN) component of the Hessian. For a datapoint (x, y), let $f(x)$ denote the output of a neural network and $\mathcal{L}(f(x), y)$ represent the training loss. Let $W \in \mathbb{R}^{m \times n}$ represent a weight matrix in the neural network and $\mathcal{D}$ denote the training distribution. Then, for CE loss, the Gauss-Newton component of the Hessian of the loss with respect to W is given by (see Appendix D for details)

$$H_{\text{GN}} = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\frac{\partial f}{\partial W}\frac{\partial^2 \mathcal{L}}{\partial f^2}\frac{\partial f}{\partial W}^\top\right] = \mathop{\mathbb{E}}_{\substack{x\sim\mathcal{D}_x \\ s\sim f(x)}}\left[g_{x,s} g_{x,s}^\top\right],$$

where, for brevity, $f(x)$ denotes the output distribution of the neural network and $\mathcal{D}_x$ represents the training distribution of x (Pascanu & Bengio, 2014). The right-hand side of the equation is also referred to in the literature as the Fisher matrix, and its counterpart for real labels, $\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[g_{x,y} g_{x,y}^\top\right]$, is referred to as the empirical Fisher. For brevity, going forward, we will assume that x is drawn from $\mathcal{D}_x$ and represent the Fisher matrix as $\mathbb{E}_{x,s\sim f(x)}\left[g_{x,s} g_{x,s}^\top\right]$. Similarly, when both x and y are used, we will assume they are drawn from $\mathcal{D}$.

The aim of algorithms such as K-FAC and Shampoo (when viewed from the Hessian perspective) is to do a layerwise Kronecker product approximation of the Fisher matrix H_{GN}. The following lemma establishes the approximation made by Shampoo:

Lemma 3 (Adapted from Gupta et al. (2018b); Anil et al. (2020))**.** *Assume that $G_{x,s}$ are matrices of rank at most r. Let $g_{x,s} = \text{vec}(G_{x,s})$. Then, for any $\epsilon > 0$,*

$$\mathop{\mathbb{E}}_{x,s\sim f(x)}\left[g_{x,s} g_{x,s}^\top\right] \preccurlyeq r\left(\mathop{\mathbb{E}}_{x,s\sim f(x)}\left[G_{x,s} G_{x,s}^\top\right]\right)^{1/2} \otimes \left(\mathop{\mathbb{E}}_{x,s\sim f(x)}\left[G_{x,s}^\top G_{x,s}\right]\right)^{1/2}. \tag{1}$$

[3] We will focus on weights structured as matrices throughout this paper.

In Lemma 2 the matrix on the left hand side is equal to H_{GN} and the right hand side represents the H_{GN} approximation made by Shampoo. However, computing this approximation at every step is expensive. So, in practice, Shampoo makes two additional approximations on top.

First, it replaces the per-input gradient by batch gradient, i.e, replaces $\mathbb{E}_{x,s\sim f(x)}[G_{x,s}G_{x,s}^{\top}]$ by $\mathbb{E}_{B,\mathbf{s}}[G_{B,\mathbf{s}}G_{B,\mathbf{s}}^{\top}]$, where B denotes the batch, $\mathbf{s}$ is the concatenation of $s \sim f(x)$ for all $(x, y) \in B$ and $G_{B,\mathbf{s}} = \frac{1}{|B|}\sum_{(x,y)\in B, s=\mathbf{s}[x]} G_{x,s}$ is the *sampled batch gradient*, with $\mathbf{s}[x]$ representing the sampled label corresponding to $x \in B$.

Second, it replaces sampled labels with real labels, i.e., it replaces $\mathbb{E}_{B,\mathbf{s}}[G_{B,\mathbf{s}}G_{B,\mathbf{s}}^{\top}]$ with $\mathbb{E}_{B}[G_{B}G_{B}^{\top}]$, where $G_B = \frac{1}{|B|}\sum_{(x,y)\in B} G_{x,y}$ is the *batch gradient*.

Thus, if G_j and W_j represent the batch gradient and weight matrix at iteration j, and λ is an exponential weighting parameter, then the update of Shampoo is given by

$$L_j = \lambda L_{j-1} + (1-\lambda)G_jG_j^{\top}; \quad R_j = \lambda R_{j-1} + (1-\lambda)G_j^{\top}G_j; \quad W_{j+1} = W_j - \eta L_j^{-1/4}G_jR_j^{-1/4},$$

where L_j and R_j represent the left and right preconditioners maintained by Shampoo, respectively.

Our focus (when viewing Shampoo from the Hessian perspective) will be to study

- The optimal Kronecker product approximation of the matrix H_{GN} and its connection to Shampoo's approximation (done in Section 3).
- The effect of the aforementioned two approximations on the approximation quality (done in Section 4).

2.2 Optimal Kronecker product approximation

For Frobenius norm (or other "entry-wise" matrix norms), finding the optimal Kronecker product approximation of a matrix $H \in \mathbb{R}^{mn\times mn}$ is equivalent to finding the optimal rank-one approximation of a rearrangement of H. We define the rearrangement operator $\text{reshape}()$, applied to a matrix H such that,

$$\text{reshape}(H)[mi + i', nj + j'] = H[mj + i, mj' + i'],$$

where $\{i, i'\} \in [0, 1, ..., m-1]$, $\{j, j'\} \in [0, 1, ..., n-1]$ and $\text{reshape}(H) \in \mathbb{R}^{m^2\times n^2}$. A property of $\text{reshape}()$ that will be useful to us is:

$$H = A \otimes B \iff \text{reshape}(H) = ab^{\top}, \tag{2}$$

where $A \in \mathbb{R}^{m\times m}$, $a = \text{vec}(A) \in \mathbb{R}^{m^2}$, $B \in \mathbb{R}^{n\times n}$ and $b = \text{vec}(B) \in \mathbb{R}^{n^2}$. This property can be used to prove the following result on optimal Kronecker product approximation:

Lemma 4 (Van Loan & Pitsianis (1993))**.** *Let $H \in \mathbb{R}^{mn\times mn}$ be a matrix and let $L \in \mathbb{R}^{m\times n}, R \in \mathbb{R}^{n\times m}$. Then, the equivalence of the Kronecker product approximation of H and the rank-one approximation of* $\text{reshape}(H)$ *is given by:*

$$\|H - L \otimes R\|_F = \|\,\text{reshape}(H) - \text{vec}(L)\,\text{vec}(R)^{\top}\|_F,$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

Since the optimal rank-1 approximation of a matrix is given by its singular value decomposition (SVD), we conclude:

Corollary 1. *Let $H \in \mathbb{R}^{mn\times mn}$. If the top singular vectors and singular value of* $\text{reshape}(H)$ *are represented by u_1, v_1 and σ_1, respectively, then the matrices $L \in \mathbb{R}^{m\times m}$ and $R \in \mathbb{R}^{n\times n}$ defined by*

$$\text{vec}(L) = \sigma_1 u_1, \quad \text{vec}(R) = v_1,$$

minimize the Frobenius norm $\|H - L \otimes R\|_F$.

Obtaining SVD by power iteration. Power iteration (Golub & Van Loan, 1996) is a well-known method for estimating the top eigenvalue of a matrix M. It can also be specialized for obtaining the top singular vectors of a matrix. The corresponding iterations for the left singular vector ℓ and the right singular vector r are given by

$$\ell_k \leftarrow Mr_{k-1}; \quad r_k \leftarrow M^{\top}\ell_{k-1}, \tag{3}$$

where k denotes the iteration number.

Cosine similarity. We will be using cosine similarity between matrices as a metric for approximation. For two matrices M_1 and M_2, this refers to $\text{Tr}(M_1 M_2^\top)/(||M_1||_F \cdot ||M_2||_F)$. A value of 1 indicates perfect alignment, while a value of 0 indicates orthogonality.

3 Optimal Kronecker product approximation and Shampoo

In this section, we will specialize the theory of Section 2.2 for finding the optimal Kronecker product approximation of a covariance matrix $H = \mathbb{E}_{g \sim \mathcal{D}_g}[gg^\top]$ for $g \in \mathbb{R}^{mn}$. Both perspectives of Shampoo described in Section 2.1 are concerned with Kronecker product approximations of H of the form $L \otimes R$ where $L \in \mathbb{R}^{m \times m}$, $R \in \mathbb{R}^{n \times n}$, but for different distributions $\mathcal{D}_g$. For the Adagrad viewpoint, with $\mathcal{D}_g$ as the uniform distribution over g_t where $1 \leqslant t \leqslant T$ refers to the gradient at timestep t, $H = H_{\text{Ada}}$. For the Hessian viewpoint, with $\mathcal{D}_g$ as the distribution over gradients with batch size 1 and with sampled labels, $H = H_{\text{GN}}$ (see Section 2.1.2 for derivation).

Since our results will hold for all distributions $\mathcal{D}_g$, we will use $\mathbb{E}[gg^\top]$ to refer to $\mathbb{E}_{g \sim \mathcal{D}_g}[gg^\top]$ to simplify notation. The main goal of this section will be to study the optimal Kronecker product approximation to such a generic matrix H, see its connection to Shampoo, and experimentally validate our results for $H = H_{\text{Ada}}$ and $H = H_{\text{GN}}$, which are described in Section 2.1.1 and 2.1.2, respectively.

Loan & Pitsianis (1993) describe an approach to find the optimal Kronecker product approximation of a matrix (with respect to the Frobenius norm). Koroko et al. (2023b) use this approach to find the optimal layer-wise Kronecker product approximation of the hessian matrix for networks without weight sharing. We will now do a general analysis which would also be applicable to neural networks with weight sharing.

Since $g \in \mathbb{R}^{mn}$, each entry of g can be described as a tuple $(i, j) \in [m] \times [n]$. Consequently, every entry of H can be represented by the tuple $((i, j), (i', j'))$. We now consider the matrix $\hat{H} \coloneqq \text{reshape}(H) \in \mathbb{R}^{m^2 \times n^2}$, which is a rearrangement (see Section 2) of the entries of H.

By using equation 2 we get that:

$$\hat{H} = \mathbb{E}[G \otimes G].$$

Further, by Lemma 4, we have that if $L \otimes R$ is the optimal Kronecker product approximation of H, then $\ell r^\top$ is the optimal rank-1 approximation of $\hat{H}$, where $\ell = \text{vec}(L)$ and $r = \text{vec}(R)$. Hence, the problem reduces to finding the optimal rank-1 approximation of $\hat{H}$. Applying the power iteration scheme described in Equation 3 for estimating the top singular vectors of $\hat{H}$ and using Lemma 1 yields (where k denotes the k^{th} step of power iteration):

$$\ell_k \leftarrow \hat{H} r_{k-1} = \mathbb{E}[G \otimes G] r_{k-1} = \text{vec}(\mathbb{E}[G R_{k-1} G^\top]),$$
$$r_k \leftarrow \hat{H}^\top \ell_{k-1} = \mathbb{E}[G \otimes G]^\top \ell_{k-1} = \text{vec}(\mathbb{E}[G^\top L_{k-1} G]).$$

Reshaping vectors on both sides into matrices results in:

$$L_k \leftarrow \mathbb{E}[G R_{k-1} G^\top]; \quad R_k \leftarrow \mathbb{E}[G^\top L_{k-1} G]. \tag{4}$$

3.1 One round of power iteration

Our first and main approximation involves replacing the iterative power iteration scheme (Equation 4) with just a single iteration. This leads to the main contribution of our work:

Proposition 1. *One step of power iteration, starting from the identity, for obtaining the optimal Kronecker product approximation of H is precisely equal to the square of the Shampoo's approximation of H*

Proof. The initialization for the single iteration will use the identity matrix, i.e., I_m and I_n for L and R, respectively. Thus, we transition from the iterative update equations:

$$L_k \leftarrow \mathbb{E}[G R_{k-1} G^\top]; \quad R_k \leftarrow \mathbb{E}[G^\top L_{k-1} G],$$

to the simplified single-step expressions:

$$L \leftarrow \mathbb{E}[GG^\top]; \quad R \leftarrow \mathbb{E}[G^\top G].$$

With the above expression for L and R, $L \otimes R$ is precisely equal to the *square* of the Shampoo's approximation of H given by the right hand side of Equation 1. □

As shown in Figure 1, for various datasets and architectures, this single step of power iteration is very close to the optimal Kronecker product approximation for both $H = H_{\text{GN}}$ (top) and $H = H_{\text{Ada}}$ (bottom). However, we can see that the upper bound proposed by the original Shampoo work (Gupta et al., 2018b) is significantly worse.

3.1.1 Why initialize with the identity matrix?

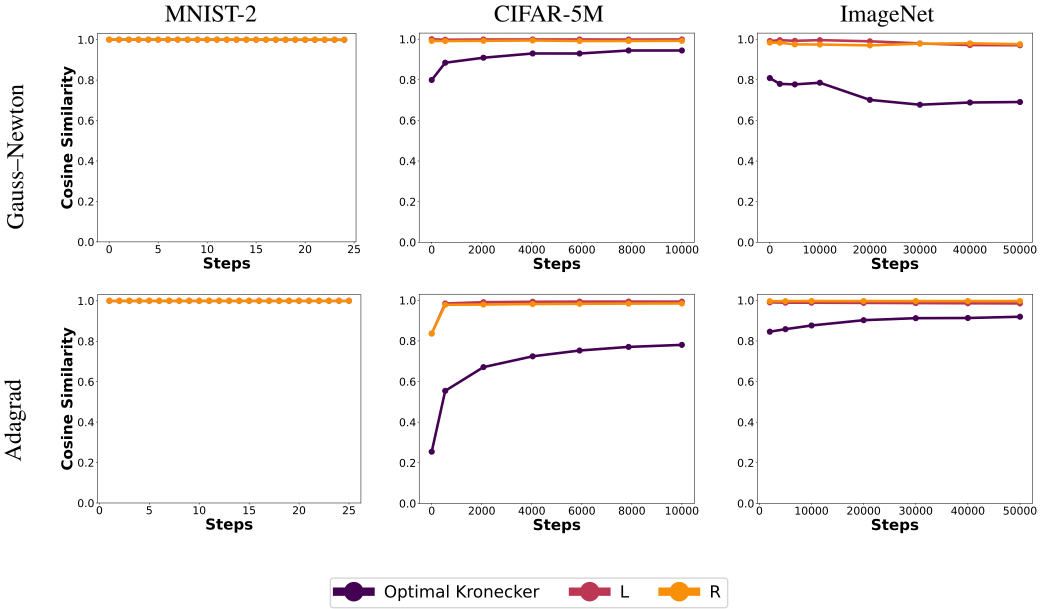

Figure 2: Comparing $\frac{\sigma_1}{\sqrt{\sum_i \sigma_i^2}}$ and $\frac{\alpha_1\sigma_1}{\sqrt{\sum_i \alpha_i^2\sigma_i^2}}$ for various datasets and architectures. The top row is for $H = H_{\text{GN}}$ while the bottom row is for $H = H_{\text{Ada}}$. The L and R legends represent $\frac{\alpha_1\sigma_1}{\sqrt{\sum_i \alpha_i^2\sigma_i^2}}$ for the left and right singular vector respectively. The "Optimal Kronecker" legend represents $\frac{\sigma_1}{\sqrt{\sum_i \sigma_i^2}}$ (see Section 3.1.1). As seen, $\frac{\alpha_1\sigma_1}{\sqrt{\sum_i \alpha_i^2\sigma_i^2}}$ is much closer to 1 as compared to $\frac{\sigma_1}{\sqrt{\sum_i \sigma_i^2}}$, demonstrating the role played by identity initialization in ensuring convergence of power iteration in one round. See Appendix B.1 for details.

Suppose the SVD of $\hat{H}$ is given by $\hat{H} = \sum_i \sigma_i u_i v_i^T$, or equivalently, $H = \sum_i \sigma_i U_i \otimes V_i$. The convergence of the power iteration in one step depends on the inner product of the initialization vector with the top singular vector. Let us focus on the left side,[4] i.e., the update $L \leftarrow \mathbb{E}[GG^\top]$ which as described earlier is equivalent to starting with the initialization I_n. Let $\text{vec}\,(I_n) = \sum_i \alpha_i v_i$ i.e. $I_n = \sum_i \alpha_i V_i$. After one iteration, we obtain $\ell := \sum_i \alpha_i\sigma_i u_i$, and correspondingly, $L := \sum_i \alpha_i\sigma_i U_i$. We are interested in assessing how closely ℓ approximates the leading eigenvector u_1. The cosine similarity between ℓ and u_1 is given by $\frac{\alpha_1\sigma_1}{\sqrt{\sum_i \alpha_i^2\sigma_i^2}}$.

One reason why the cosine similarity might be large is that $\hat{H}$ is nearly rank-1 (σ_1 is large); that is, H is closely approximated by a Kronecker product. As illustrated in Figure 1, this assumption does not universally hold. Instead, we propose an alternative explanation for why a single step of power

[4] The discussion for the other side is analogous.

iteration is typically sufficient: the coefficient α_1 is usually larger than α_i for all $i \geqslant 2$. We begin by providing a theoretical justification for this, followed by empirical evidence from our experiments.

We start by noting that $\alpha_i = \operatorname{vec}(I_n)^T v_i = \operatorname{Tr}(V_i)$. Now, we will show that using the identity matrix as initialization is a good choice since a) shows it has the maximal dot product with possible top components i.e., PSD matrices (Proposition 2), and b) we expect it to have a small dot product with later components.

Lemma 5 (Loan & Pitsianis (1993)). *V_1 is a Positive Semi-Definite (PSD) matrix.*

Since V_1 is a PSD matrix we would like to initialize our power iteration with a matrix which is close to all PSD matrices. Now, we will show that identity is the matrix which achieves this, specifically it maximizes the minimum dot product across the set of PSD matrices of unit Frobenius norm.

Proposition 2. *Consider the set of PSD matrices of unit Frobenius norm of dimension m denoted by S_m. Then*

$$\frac{1}{\sqrt{m}} I_m = \underset{M \in S_m}{\arg\max} \min_{M' \in S_m} \langle \operatorname{vec}(M), \operatorname{vec}(M') \rangle.$$

The previous proposition argues that I_m maximizes the worst-case dot product with possible top singular vectors. Now, we argue that its dot product with other singular vectors should be lower.

Lemma 6. *If V_1 is positive-definite, then V_i for $i \geqslant 2$ are not PSD.*

Therefore, the diagonal elements of V_i for $i \geqslant 2$ need not be positive, and this might lead to cancellations (for $i \geqslant 2$) in the trace of V_i which is equal to α_i. Hence we expect α_i's for $i \geqslant 2$ to be smaller than α_1. We now show experiments to demonstrate this in practice. To quantify the benefit of α_1 usually being larger than α_i for $i \geqslant 2$, we will compare $\frac{\alpha_1 \sigma_1}{\sqrt{\sum_i \alpha_i^2 \sigma_i^2}}$ (for both left and right singular vectors) and $\frac{\sigma_1}{\sqrt{\sum_i \sigma_i^2}}$. The latter can be interpreted as the cosine similarity if all α's were equal or as a measure of how close $\hat{H}$ is to being rank 1 since it is equal to the cosine similarity between $u_1 v_1^T$ and $\hat{H}$. Thus $\frac{\sigma_1}{\sqrt{\sum_i \sigma_i^2}}$ is equal to the "Optimal Kronecker" cosine similarity used in Figure 1. In Figure 2 we track both of these quantities through training and indeed observe that $\frac{\alpha_1 \sigma_1}{\sqrt{\sum_i \alpha_i^2 \sigma_i^2}}$ are significantly closer to 1 than $\frac{\sigma_1}{\sqrt{\sum_i \sigma_i^2}}$ for both $H = H_{\text{GN}}$ (top) and $H = H_{\text{Ada}}$ (bottom).

3.1.2 Exact Kronecker product structure in H

The previous discussion shows that $\mathbb{E}\left[GG^\top\right] \otimes \mathbb{E}\left[G^\top G\right]$ is close to the optimal Kronecker product approximation of H. In this section we will show that this holds exactly if H is a Kronecker product. Intuitively, this holds since if H is a Kronecker product, then $\hat{H}$ is rank-1, and one round of power iteration would recover $\hat{H}$. Until now, we have been focusing on the direction of top singular vectors of $\hat{H}$, but with the assumption of $\hat{H}$ being rank 1, we can compute the explicit expression for $\hat{H}$, and hence of H.

Corollary 2. *Under the assumption that $\hat{H}$ is rank-1,*

$$H = \left(\mathbb{E}\left[GG^\top\right] \otimes \mathbb{E}\left[G^\top G\right]\right) / \operatorname{Tr}\left(\mathbb{E}\left[GG^\top\right]\right).$$

Proof. Let $\hat{H} = \sigma u v^\top$, i.e, $H = \sigma U \otimes V$. Let $I_m = \operatorname{Tr}(U)U + R_m$ and $I_n = \operatorname{Tr}(V)V + R_n$, where R_m and R_n are the residual matrices. Now, after one round of power iteration, the left and right estimates provided by Shampoo are given by

$$\mathbb{E}\left[GG^\top\right] = \sigma \operatorname{Tr}(V)U, \quad \mathbb{E}\left[G^\top G\right] = \sigma \operatorname{Tr}(U)V.$$

From this, we can see that $\operatorname{Tr}\left(\mathbb{E}\left[GG^\top\right]\right) = \sigma \operatorname{Tr}(U) \operatorname{Tr}(V)$. Thus

$$H = \sigma U \otimes V = \left(\mathbb{E}\left[GG^\top\right] \otimes \mathbb{E}\left[G^\top G\right]\right) / \operatorname{Tr}\left(\mathbb{E}\left[GG^\top\right]\right).$$

□

Since $H = \hat{H}_{\text{GN}}$ is an $m^2 \times 1$ matrix for binomial logistic regression, it is rank-1, so the equality in the corollary holds. In other words, the square of Shampoo's H_{GN} estimate perfectly correlates with H_{GN} for binomial logistic regression. This is demonstrated in the first plot of Figure 1.

We note that $\left(\mathbb{E}\left[GG^\top\right] \otimes \mathbb{E}\left[G^\top G\right]\right) / \operatorname{Tr}\left(\mathbb{E}\left[GG^\top\right]\right)$ as an estimate of H was also derived by Ren & Goldfarb (2021). But their assumptions were much stronger than ours, specifically they assume that the gradients follow a *tensor-normal distribution*, which implies that $\hat{H}$ is rank 1. Instead, we only make a second moment assumption on the gradients: $H = \mathbb{E}[gg^\top]$ is an exact Kronecker product. We also note that our derivation of the *direction* $\mathbb{E}\left[GG^\top\right] \otimes \mathbb{E}\left[G^\top G\right]$ being close to the optimal Kronecker product approximation holds independently of $\hat{H}$ being rank 1.

3.1.3 Discussion about optimization

Let us refer to $\mathbb{E}[GG^\top] \otimes \mathbb{E}[G^\top G]$ by H_1. As mentioned in Equation 1, the original Shampoo paper used the approximation H used was $H_{1/2} \coloneqq \mathbb{E}[GG^\top]^{1/2} \otimes \mathbb{E}[G^\top G]^{1/2}$. In practice, when using Shampoo as an optimization algorithm, the gradient step is taken in the direction of $H_{1/2}^{-p}\nabla L$ where p is tuned as a hyperparameter (Anil et al., 2020; Shi et al., 2023). Since $H_{1/2}^{-p} = H_1^{-p/2}$, searching over p in $H_{1/2}^{-p}$ yields the same search space as H_1^{-p}. Therefore, the difference between H_1 and $H_{1/2}$ does not manifest practically in optimization speed, but it yields a significant difference in our understanding of how Shampoo works.

4 Hessian Approximation of Shampoo

From the Hessian approximation viewpoint, the previous section covers the case of using batch size 1 and sampled labels, as described in Section 2.1.2. To be precise, in Figure 1 top, we consider how well H_{GN} is correlated with $E_{x,s}[G_{x,s}G_{x,s}^T] \otimes E_{x,s}[G_{x,s}^T G_{x,s}]$, where s represents that the labels are sampled from the model's output distribution. On the other hand, as discussed in Section 2.1.2, Shampoo in practice is generally used with arbitrary batch sizes and real labels. We now investigate the effect of these two factors on the Hessian approximation.

4.1 Averaging gradients across the batch

The next approximation towards Shampoo is to average the gradient across the batch, i.e., we go from

$$L \leftarrow \mathop{\mathbb{E}}_{x,s\sim f(x)}[G_{x,s}G_{x,s}^\top]; \quad R \leftarrow \mathop{\mathbb{E}}_{x,s\sim f(x)}[G_{x,s}^\top G_{x,s}]$$

to

$$L \leftarrow |B| \mathop{\mathbb{E}}_{B,\mathbf{s}}[G_{B,\mathbf{s}}G_{B,\mathbf{s}}^\top]; \quad R \leftarrow |B| \mathop{\mathbb{E}}_{B,\mathbf{s}}[G_{B,\mathbf{s}}^\top G_{B,\mathbf{s}}],$$

where B denotes the batch, $\mathbf{s}$ is the concatenation of $s \sim f(x)$ for all $x \in B$ and $G_{B,\mathbf{s}} = \frac{1}{|B|}\sum_{x\in B, s=\mathbf{s}[x]} G_{x,s}$ is the batch gradient, with $\mathbf{s}[x]$ representing the sampled label corresponding to $x \in B$.

As previous works have shown, this change does not have any effect in expectation due to $G_{x,s}$ being mean zero for all x when we take expectation over $s \sim f(x)$ (Bartlett, 1953) i.e. $\mathbb{E}_s[G_{x,s}] = 0$.

Lemma 7 (Implicitly in Liu et al. (2024); Osawa et al. (2023b))**.**

$$|B| \mathop{\mathbb{E}}_{B,\mathbf{s}}[G_{B,\mathbf{s}}G_{B,\mathbf{s}}^\top] = \mathop{\mathbb{E}}_{x,s\sim f(x)}[G_{x,s}G_{x,s}^\top].$$

However, this does lead to a significant improvement in computational complexity by saving up to a factor of batch size.

4.2 Using real labels instead of sampled labels

As our final approximation we replace using sampled labels $s \sim f(x)$ to using real labels y. This approximation, denoted in the literature by empirical Fisher when batch size is 1, has been discussed at length by prior works (Osawa et al., 2023a; Kunstner et al., 2019). The main theoretical argument

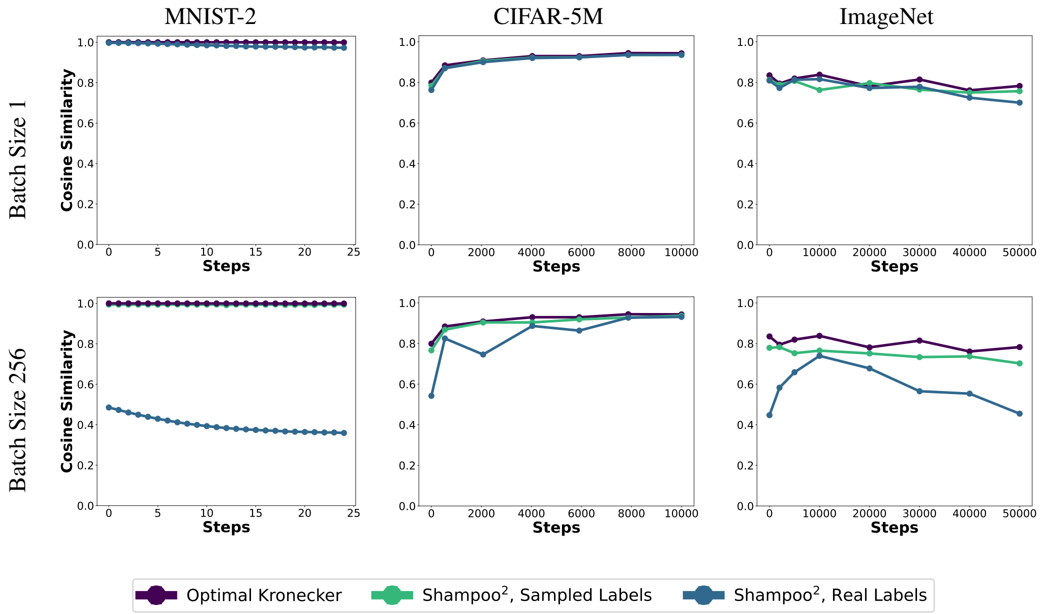

Figure 3: Cosine similarity between approximations of H_{GN} and its true value. First row is for batch size 1 while the second row is for batch size 256. We observe deterioration in approximation quality at larger batch size. We note that the batch size does not refer to the batch size used in optimization, rather it refers to the batch size used for Hessian approximation.

for why this approximation may work well is that, as we move towards optima, the two quantities converge in the presence of label noise (Grosse, 2021).

In Figure 3 (top), when evaluating H_{GN} approximation with batch size 1, we surprisingly find that the approximation quality is good throughout the training. However, unlike the case of sampled labels, the approximation starts to degrade at large batch sizes because the gradients with real labels are not mean 0. The lemma below (Grosse, 2021) shows how this estimator changes with batch size.

Lemma 8 (Grosse (2021))**.** *Let B denote the batch and $G_B = \frac{1}{|B|}\sum_{(x,y)\in B} G_{x,y}$ denote the batch gradient. Then*

$$\mathop{\mathbb{E}}_{B}[G_B G_B^\top] = \frac{1}{|B|}\mathop{\mathbb{E}}_{x,y}[G_{x,y}G_{x,y}^\top] + \left(1 - \frac{1}{|B|}\right)\mathop{\mathbb{E}}_{x,y}[G_{x,y}]\mathop{\mathbb{E}}_{x,y}[G_{x,y}]^\top.$$

The above lemma shows that, depending on the batch size, the estimator interpolates between $\mathbb{E}_{x,y}[G_{x,y}G_{x,y}^\top]$ (Empirical Fisher) and $\mathbb{E}_{x,y}[G_{x,y}]\,\mathbb{E}_{x,y}[G_{x,y}]^\top$. As shown in Figure 3 (top), at batch size 1, when $\mathbb{E}_B[G_B G_B^\top]$ is equal to $\mathbb{E}_{x,y}[G_{x,y}G_{x,y}^\top]$, it closely tracks the optimal Kronecker product approximation. In other words, approximating the empirical Fisher is nearly sufficient in our experiments to recover the optimal Kronecker product approximation to H_{GN}. However, with increasing batch size (Figure 3, bottom row), the approximation quality degrades.

We note that this approximation has the computational benefit of not requiring another backpropagation with sampled labels; instead, these computations can be done alongside usual training.

5 Related work

We discuss the related works in detail in Appendix E. Here, we discuss two closely related works: Ren & Goldfarb (2021) and Koroko et al. (2023a).

Ren & Goldfarb (2021) study the Hessian perspective of Shampoo and show that, under the assumption that sampled gradients follow a *tensor-normal* distribution, the square of the Hessian estimate of Shampoo is perfectly correlated with H_{GN}. We also show the same result under much weaker conditions in Corollary 2. Moreover, in Proposition 1 we show that, in general, the square of the

Hessian estimate of Shampoo is closely related to the optimal Kronecker product approximation of H_{GN}. We additionally also study the approximations used by Shampoo to make it computationally efficient (Section 4) and the Adagrad perspective of Shampoo's preconditioner.

Loan & Pitsianis (1993) develop the theory of optimal Kronecker product approximation of a matrix (in Frobenius norm). Koroko et al. (2023a) use it for finding layer-wise optimal Kronecker product approximation of H_{GN} for a network without weight sharing. We extend their technique to networks with weight-sharing, and show that the square of the Hessian estimate of Shampoo is nearly equivalent to the optimal Kronecker product approximation of H_{GN}.

6 Limitations

The main contribution of our work is to show that the square of the Shampoo's approximation of H (where H refers to either H_{Ada} or H_{GN}) is nearly equivalent to the optimal Kronecker approximation of H. Although we verify this empirically on various datasets and provide theoretical arguments, the gap between them depends on the problem structure. In some of our experiments with ViT architecture (Appendix A), we find that the gap is relatively larger compared to other architectures. Moreover, it remains an open question to understand the conditions (beyond those described in K-FAC Martens & Grosse (2015b)) under which H is expected to be close to a Kronecker product. Again, in some of the experiments with ViTs (Appendix A), we find that the optimal Kronecker product approximation to H is much worse as compared to other architectures.

Acknowledgements

NV and DM are supported by a Simons Investigator Fellowship, NSF grant DMS-2134157, DARPA grant W911NF2010021, and DOE grant DE-SC0022199. This work has been made possible in part by a gift from the Chan Zuckerberg Initiative Foundation to establish the Kempner Institute for the Study of Natural and Artificial Intelligence. SK and DM acknowledge funding from the Office of Naval Research under award N00014-22-1-2377 and the National Science Foundation Grant under award #IIS 2229881. LJ acknowledges funding from the National Science Foundation DMS-2134157.

References

Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. Towards practical second order optimization for deep learning. 2020.

Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. Scalable second order optimization for deep learning, 2021.

Rohan Anil, Sandra Gadanho, Da Huang, Nijith Jacob, Zhuoshu Li, Dong Lin, Todd Phillips, Cristina Pop, Kevin Regan, Gil I Shamir, et al. On the factory floor: Ml engineering for industrial-scale ads recommendation models. *arXiv preprint arXiv:2209.05310*, 2022.

Lukas Balles, Fabian Pedregosa, and Nicolas Le Roux. The geometry of sign gradient descent, 2020.

M. S. Bartlett. Approximate confidence intervals. *Biometrika*, 40(1/2):12–19, 1953. ISSN 00063444.

Minhyung Cho, Chandra Dhir, and Jaehyung Lee. Hessian-free optimization for learning deep multidimensional recurrent neural networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

George E. Dahl, Frank Schneider, Zachary Nado, Naman Agarwal, Chandramouli Shama Sastry, Philipp Hennig, Sourabh Medapati, Runa Eschenhagen, Priya Kasimbeg, Daniel Suo, Juhan Bae, Justin Gilmer, Abel L. Peirson, Bilal Khan, Rohan Anil, Mike Rabbat, Shankar Krishnan, Daniel Snider, Ehsan Amid, Kongtao Chen, Chris J. Maddison, Rakshith Vasudev, Michal Badura, Ankush Garg, and Peter Mattson. Benchmarking neural network training algorithms, 2023.

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*, pp. 248–255. Ieee, 2009.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011a.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011b.

Sai Surya Duvvuri, Fnu Devvrit, Rohan Anil, Cho-Jui Hsieh, and Inderjit S Dhillon. Combining axes preconditioners through kronecker approximation for deep learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Runa Eschenhagen, Alexander Immer, Richard E Turner, Frank Schneider, and Philipp Hennig. Kronecker-factored approximate curvature for modern neural network architectures. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Kai-Xin Gao, Xiao-Lei Liu, Zheng-Hai Huang, Min Wang, Shuangling Wang, Zidong Wang, Dachuan Xu, and Fan Yu. Eigenvalue-corrected natural gradient based on a new approximation, 2020.

Kaixin Gao, Xiaolei Liu, Zhenghai Huang, Min Wang, Zidong Wang, Dachuan Xu, and Fan Yu. A trace-restricted kronecker-factored approximation to natural gradient. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(9):7519–7527, May 2021. doi: 10.1609/aaai.v35i9.16921. URL `https://ojs.aaai.org/index.php/AAAI/article/view/16921`.

Jezabel R Garcia, Federica Freddi, Stathi Fotiadis, Maolin Li, Sattar Vakili, Alberto Bernacchia, and Guillaume Hennequin. Fisher-legendre (fishleg) optimization of deep neural networks. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=c9lAOPvQHS`.

Google Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. `https://storage.googleapis.com/deepmind-media/gemini/gemini_v1_5_report.pdf`, 20024. [Online; accessed 19-May-2024].

Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. Fast approximate natural gradient descent in a kronecker factored eigenbasis. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper_files/paper/2018/file/48000647b315f6f00f913caa757a70b3-Paper.pdf`.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.

Roger Grosse. Adaptive gradient methods, normalization, and weight decay. `https://www.cs.toronto.edu/~rgrosse/courses/csc2541_2021/readings/L05_normalization.pdf`, 2021.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1842–1850. PMLR, 10–15 Jul 2018a. URL `https://proceedings.mlr.press/v80/gupta18a.html`.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pp. 1842–1850. PMLR, 2018b.

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

Harold V Henderson and Shayle R Searle. The vec-permutation matrix, the vec operator and kronecker products: A review. *Linear and multilinear algebra*, 9(4):271–288, 1981.

Abdoulaye Koroko, Ani Anciaux-Sedrakian, Ibtihel Gharbia, Valérie Garès, Mounir Haddou, and Quang Huy Tran. Efficient approximations of the fisher matrix in neural networks using kronecker product singular value decomposition. *ESAIM: Proceedings and Surveys*, 73:218–237, 2023a. doi: 10.1051/proc/202373218. URL `https://hal.science/hal-04266143`.

Abdoulaye Koroko, Ani Anciaux-Sedrakian, Ibtihel Ben Gharbia, Valérie Garès, Mounir Haddou, and Quang Huy Tran. Efficient approximations of the fisher matrix in neural networks using kronecker product singular value decomposition. *ESAIM: Proceedings and Surveys*, 73:218–237, 2023b.

Frederik Kunstner, Philipp Hennig, and Lukas Balles. Limitations of the empirical fisher approximation for natural gradient descent. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper_files/paper/2019/file/46a558d97954d0692411c861cf78ef79-Paper.pdf`.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Wu Lin, Felix Dangel, Runa Eschenhagen, Juhan Bae, Richard E. Turner, and Alireza Makhzani. Can we remove the square-root in adaptive gradient methods? a second-order perspective. arXiv 2402.03496, 2024.

Hong Liu, Zhiyuan Li, David Leo Wright Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *The Twelfth International Conference on Learning Representations*, 2024.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11976–11986, June 2022.

C. F. Van Loan and N. Pitsianis. Approximation with kronecker products. In Bart L. R. Moor Marc S. Moonen, Gene H. Golub (ed.), *Linear Algebra for Large Scale and Real-Time Applications*, pp. 293–314. Springer, 1993.

James Martens. Deep learning via hessian-free optimization. In Johannes Fürnkranz and Thorsten Joachims (eds.), *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pp. 735–742. Omnipress, 2010. URL https://icml.cc/Conferences/2010/papers/458.pdf.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2408–2417, Lille, France, 07–09 Jul 2015a. PMLR. URL https://proceedings.mlr.press/v37/martens15.html.

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

James Martens and Ilya Sutskever. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pp. 1033–1040, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

James Martens, Jimmy Ba, and Matt Johnson. Kronecker-factored curvature approximations for recurrent neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HyMTkQZAb.

James Martens et al. Deep learning via hessian-free optimization. In *Icml*, volume 27, pp. 735–742, 2010.

Preetum Nakkiran, Behnam Neyshabur, and Hanie Sedghi. The deep bootstrap framework: Good online learners are good offline generalizers. *arXiv preprint arXiv:2010.08127*, 2020.

Kazuki Osawa, Yohei Tsuji, Yuichiro Ueno, Akira Naruse, Rio Yokota, and Satoshi Matsuoka. Large-scale distributed second-order optimization using kronecker-factored approximate curvature for deep convolutional neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12351–12359, 2019. doi: 10.1109/CVPR.2019.01264.

Kazuki Osawa, Satoki Ishikawa, Rio Yokota, Shigang Li, and Torsten Hoefler. ASDL: A unified interface for gradient preconditioning in pytorch. *CoRR*, abs/2305.04684, 2023a. doi: 10.48550/ARXIV.2305.04684. URL https://doi.org/10.48550/arXiv.2305.04684.

Kazuki Osawa, Satoki Ishikawa, Rio Yokota, Shigang Li, and Torsten Hoefler. Asdl: A unified interface for gradient preconditioning in pytorch, 2023b.

Vardan Papyan. Measurements of three-level hierarchical structure in the outliers in the spectrum of deepnet hessians. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5012–5021. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/papyan19a.html.

Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. In Yoshua Bengio and Yann LeCun (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1301.3584.

Yi Ren and Donald Goldfarb. Tensor normal training for deep learning models. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 26040–26052. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/dae3312c4c6c7000a37ecfb7b0aeb0e4-Paper.pdf.

Adepu Ravi Sankar, Yash Khasbage, Rahul Vigneswaran, and Vineeth N Balasubramanian. A deeper look at the hessian eigenspectrum of deep neural networks and its applications to regularization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(11):9481–9488, May 2021. doi: 10.1609/aaai.v35i11.17142. URL https://ojs.aaai.org/index.php/AAAI/article/view/17142.

Hao-Jun Michael Shi, Tsung-Hsien Lee, Shintaro Iwasaki, Jose Gallego-Posada, Zhijing Li, Kaushik Rangadurai, Dheevatsa Mudigere, and Michael Rabbat. A distributed data-parallel pytorch implementation of the distributed shampoo optimizer for training neural networks at-scale, 2023.

Charles F Van Loan and Nikos Pitsianis. *Approximation with Kronecker products*. Springer, 1993.

A Additional experimental results

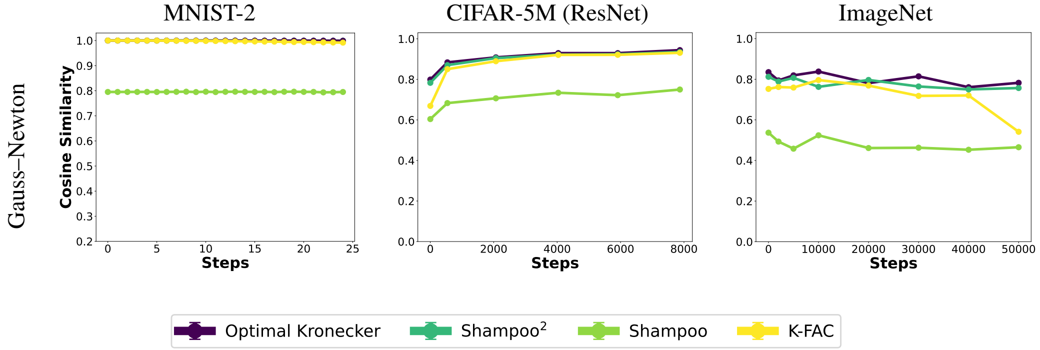

Figure 4: Cosine similarity between different approximations of the Gauss–Newton (GN) component of the Hessian and its true value for different datasets and architectures. As can be seen, Shampoo^2 tracks the optimal Kronecker approximation much more closely than Shampoo. These plots also include the K-FAC approximation, and we note that Shampoo^2 always outperforms K-FAC, though they are close in some settings.

A.1 ViT architecture

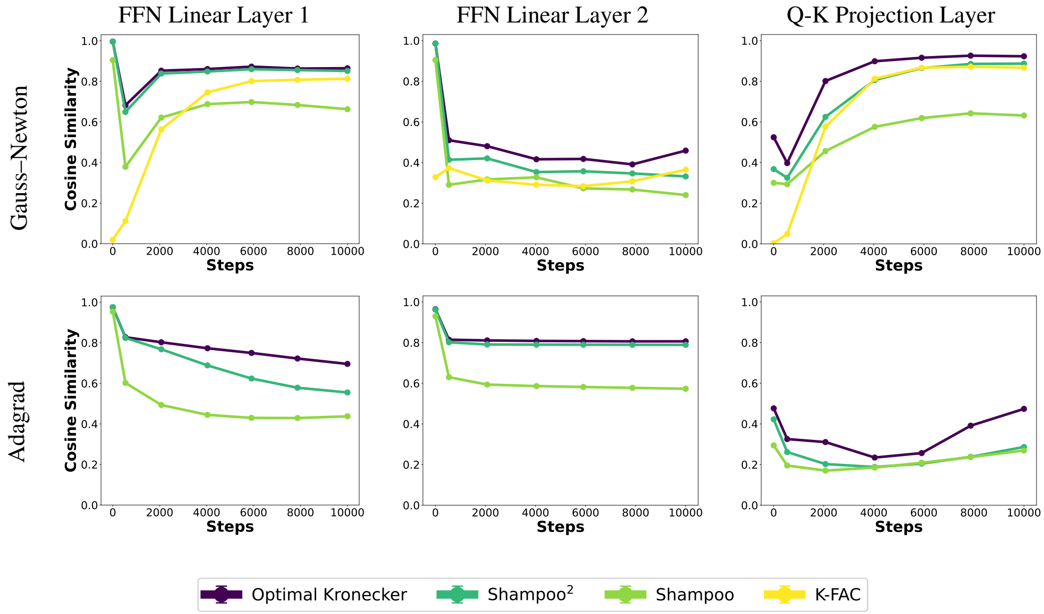

Figure 5: Analogue of Figure 1 for ViT architecture and the CIFAR-5m dataset for 3 layers of the network. For some of the figures we observe relatively larger gaps between Shampoo^2 and optimal Kronecker approximation.

In this subsection, we present the results for a Vision Transformer (ViT) architecture trained on the CIFAR-5m dataset. This architecture features a patch size of 4, a hidden dimension of 512, an MLP dimension of 512, 6 layers, and 8 attention heads.

For these experiments, we utilize three layers from the fourth transformer block: two layers from the MLP (referred to as ’FFN Linear Layer 1’ and ’FFN Linear Layer 2’) and the QK layer[5] (referred to as ’Q-K Projection Layer’).

[5] The QK layer is separated from the V part of the layer, following similar decomposition method described by Duvvuri et al. (2024)

B Experiments

Datasets and Architectures. We conducted experiments on three datasets: MNIST (LeCun et al., 1998), CIFAR-5M (Nakkiran et al., 2020), and ImageNet (Deng et al., 2009), using logistic regression, ResNet18 (He et al., 2016), and ConvNeXt-T (Liu et al., 2022) architectures, respectively. For MNIST, we subsampled two digits ($\{0, 1\}$) and trained a binary classifier.

Table 1: Summary of Experimental Configurations. λ denotes weight decay and β_1 indicates momentum.

Dataset	Architecture	Optimizer	Batch Size	Steps	lr	λ	β_1
MNIST	Linear Classifier	GD	Full Batch	25	0.01	None	0
CIFAR-5M	ResNet18	SGD	128	10000	.02	None	.9
ImageNet	ConvNeXt-T	AdamW	2048	50000	3e-3	5e-3	0.9

For MNIST, we used the only layer, i.e, the first layer of the linear classifier for computing the cosine similarities. For Resnet18 and Imagenet, we picked arbitrary layers. In particular, for Resnet 18, we used one of the convolution layers within the first block ('layer1.1.conv1' in `https://pytorch.org/vision/master/_modules/torchvision/models/resnet.html#resnet18`). For Imagenet, we used the 1x1 convolutional layer within the 2nd block of convnext-T ('stages.2.1.pwconv1' in `https://pytorch.org/vision/main/models/generated/torchvision.models.convnext_tiny.html#torchvision.models.convnext_tiny`).

Cosine similarity estimation for H_{GN}. For estimating the Frobenius norm of H_{GN}, we used the identity:

$$\mathop{\mathbb{E}}_{v\sim\mathcal{N}(0,I_d)}[v^\top H_{\text{GN}}^2 v] = \mathop{\mathbb{E}}_{v\sim\mathcal{N}(0,I_d)}[\|H_{\text{GN}}v\|_2^2] = \|H_{\text{GN}}\|_F^2$$

Hessian-vector products with the Gauss–Newton component were performed using the DeepNetHessian library provided by Papyan (2019).

For estimating the cosine similarity between H_{GN} and its estimator $\widetilde{H}_{\text{GN}}$, we used the following procedure:

1. Estimate $\|H_{\text{GN}}\|_F$, and calculate $\|\widetilde{H}_{\text{GN}}\|_F$.
2. Define scaled $\widetilde{H}_{\text{GN}}$ as $\widetilde{S}_{\text{GN}} = \frac{\|H_{\text{GN}}\|_F}{\|\widetilde{H}_{\text{GN}}\|_F}\widetilde{H}_{\text{GN}}$.
3. Cos-sim$(H_{\text{GN}}, \widetilde{H}_{\text{GN}}) = 1 - \frac{\|H_{\text{GN}}-\widetilde{S}_{\text{GN}}\|_F^2}{2\|H_{\text{GN}}\|_F^2}$, where the numerator is again estimated via Hessian-vector products.

Note that in the above procedure, we can exactly calculate $\|\widetilde{H}_{\text{GN}}\|_F$ as it is generally of a Kronecker product form with both terms of size $m \times m$ or $n \times n$, where $m \times n$ is the size of a weight matrix.

Cosine similarity estimation for H_{Ada}. We follow a similar recipe as before, but using a difference method for computing the product $H_{\text{Ada}}v$. For a given time T, $H_{\text{Ada}} = \sum_{t=1}^T g_t g_t^\top$. Thus, $H_{\text{Ada}}v = \sum_{t=1}^T (g_t^\top v) g_t$. We maintain this by keeping a running estimate of the quantity for multiple random vectors v during a training run, and use it for estimating the product $H_{\text{Ada}}v$.

B.1 Figure details

Optimal Kronecker method, wherever used was computed with five rounds of power iteration, starting from the identity. For $H = H_{\text{GN}}$, the Hessian approximations *Shampoo*^{2}, *Shampoo*, and *K-FAC* were done using sampled labels and a batch size of 1. For $H = H_{\text{Ada}}$ and step t, we used gradient enocoutered during the training run in steps $\leqslant t$.

K-FAC was computed with the "reduce" variant from Eschenhagen et al. (2023).

In Figure 2, the *Optimal Kronecker* legend represents the cosine similarity between the optimal Kronecker approximation of H_{GN} and H_{GN}. This is precisely equal to $\frac{\sigma_1}{\sqrt{\sum_i \sigma_i^2}}$. Similarly, the label

L (resp. *R*) represents the cosine similarity between the top left (resp. right) singular vector of $\hat{H}_{\text{GN}}$ and the estimate obtained after one round of power iteration starting from I_n (resp. I_m). This is precisely equal to $\frac{\alpha_1\sigma_1}{\sqrt{\sum_i \alpha_i^2\sigma_i^2}}$.

In Figure 3 (top), the Hessian approximation is calculated with batch size 1, i.e, $|B| = 1$ in Section 4.2. Similarly, in Figure 3 (bottom), $|B| = 256$.

C Deferred proofs

Lemma 6. *If V_1 is positive-definite, then V_i for $i \geqslant 2$ are not PSD.*

Proof. Consider two PSD matrices M_1 and M_2 having the eigenvalue decomposition $M_1 = \sum \lambda_{1i} q_{1i} q_{1i}^\top$ and $M_2 = \sum \lambda_{2i} q_{2i} q_{2i}^\top$. Then

$$\text{Tr}(M_1 M_2) = \sum_{i,j} \lambda_{1i}\lambda_{2j} \left(q_{1i}^\top q_{2j}\right)^2$$

Thus, if M_1 and M_2 have unit frobenius norm and M_1 is positive definite, then $\text{Tr}(M_1 M_2) > 0$.

Thus, if V_1 is positive definite, then by orthogonality of successive singular vectors, V_i for $i \geqslant 2$ cannot be positive semi-definite. □

Proposition 2. *Consider the set of PSD matrices of unit Frobenius norm of dimension m denoted by S_m. Then*

$$\frac{1}{\sqrt{m}} I_m = \arg\max_{M \in S_m} \min_{M' \in S_m} \langle \text{vec}(M), \text{vec}(M') \rangle.$$

Proof. Consider the eigendecomposition of any $M \in S_q$ given by $\sum_{i=1}^q \lambda_i v_i v_i^\top$. Denote $L = \{i : \lambda_i \leqslant \frac{1}{\sqrt{q}}\}$. As $\sum \lambda_i^2 = 1$, therefore, $|A| \geqslant 1$. Consider any $j \in A$. Then

$$\langle Vec(M), Vec(v_j v_j^\top) \rangle \leqslant \frac{1}{\sqrt{q}}$$

As v_j is orthogonal to the other eigenvectors. Thus, we can see

$$\max_{M \in S_q} \min_{M' \in S_q} \langle \text{vec}(M), \text{vec}(M') \rangle \leqslant \frac{1}{\sqrt{q}}$$

Moreover, for the matrix $\frac{1}{\sqrt{q}} I_q$, for any matrix M',

$$\frac{1}{\sqrt{q}} \langle I_q, M' \rangle = \frac{\text{tr}(M')}{\sqrt{q}}$$

where $\text{tr}(M')$ denotes the trace of the matrix M'. However, we know $\text{tr}(M') = \sum \lambda_i \geqslant 1$ as $\sum \lambda_i^2 = 1$. Thus

$$\frac{1}{\sqrt{q}} \langle I_q, M' \rangle = \frac{\text{tr}(M')}{\sqrt{q}} \geqslant \frac{1}{\sqrt{q}}$$

Note that this is the only matrix with this property as any other matrix will at least have one eigenvalue less than $\frac{1}{\sqrt{q}}$. Thus

$$\frac{1}{\sqrt{q}} I_q = \arg\max_{M \in S_q} \min_{M' \in S_q} \langle \text{vec}(M), \text{vec}(M') \rangle$$

□

Lemma 7 (Implicitly in Liu et al. (2024); Osawa et al. (2023b))**.**

$$|B| \mathop{\mathbb{E}}_{B,\mathbf{s}}[G_{B,\mathbf{s}} G_{B,\mathbf{s}}^\top] = \mathop{\mathbb{E}}_{x,s\sim f(x)}[G_{x,s} G_{x,s}^\top].$$

Proof. Evaluating $G_{B,\mathbf{s}} G_{B,\mathbf{s}}^T$, we get

$$G_{B,\mathbf{s}} G_{B,\mathbf{s}}^T = \frac{1}{|B|^2} \sum_{\substack{x,x' \in B, \\ s=\mathbf{s}[x], s'=\mathbf{s}[x']}} G_{x,s} G_{x',s'}^\top$$

Taking the expectation over $\mathbf{s}$ for a given B, and by using $\mathbb{E}_s[G_{x,s}] = 0$ we get

$$\mathop{\mathbb{E}}_{\mathbf{s}}[G_{B,\mathbf{s}} G_{B,\mathbf{s}}^T] = \frac{1}{|B|^2} \sum_x \mathop{\mathbb{E}}_{s\sim f(x)}[G_{x,s} G_{x,s}^\top] = \frac{1}{|B|} \mathop{\mathbb{E}}_{x\sim B, s\sim f(x)}[G_{x,s} G_{x,s}^\top]$$

Now taking an expectation over batches, we get

$$|B| \mathop{\mathbb{E}}_{B,\mathbf{s}}[G_{B,\mathbf{s}} G_{B,\mathbf{s}}^T] = \mathop{\mathbb{E}}_{x,s\sim f(x)}[G_{x,s} G_{x,s}^T]$$

□

Lemma 8 (Grosse (2021))**.** *Let B denote the batch and $G_B = \frac{1}{|B|} \sum_{(x,y)\in B} G_{x,y}$ denote the batch gradient. Then*

$$\mathop{\mathbb{E}}_{B}[G_B G_B^\top] = \frac{1}{|B|} \mathop{\mathbb{E}}_{x,y}[G_{x,y} G_{x,y}^\top] + \left(1 - \frac{1}{|B|}\right) \mathop{\mathbb{E}}_{x,y}[G_{x,y}] \mathop{\mathbb{E}}_{x,y}[G_{x,y}]^\top.$$

Proof. Evaluating $G_B G_B^T$, we get

$$G_B G_B^T = \frac{1}{|B|^2} \sum_{(x,y),(x',y')\in B} G_{x,y} G_{x',y'}^\top$$

Taking the expectation over B on both the sides, we get

$$\begin{aligned} \mathop{\mathbb{E}}_{B}\left[G_B G_B^T\right] &= \frac{1}{|B|^2} \left[|B| \mathop{\mathbb{E}}_{x,y}[G_{x,y} G_{x,y}^\top] + (|B|^2 - |B|) \mathop{\mathbb{E}}_{x,y}[G_{x,y}] \mathop{\mathbb{E}}_{x,y}[G_{x,y}]^\top \right] \\ \implies \mathop{\mathbb{E}}_{B}\left[G_B G_B^T\right] &= \frac{1}{|B|} \mathop{\mathbb{E}}_{x,y}[G_{x,y} G_{x,y}^\top] + \left(1 - \frac{1}{|B|}\right) \mathop{\mathbb{E}}_{x,y}[G_{x,y}] \mathop{\mathbb{E}}_{x,y}[G_{x,y}]^\top \end{aligned}$$

□

D Technical Background on Hessian

Gauss–Newton (GN) component of the Hessian. For a datapoint (x, y), let $f(x)$ denote the output of a neural network and $\mathcal{L}(f(x), y)$ represent the training loss. Let $W \in \mathbb{R}^{m\times n}$ represent a weight matrix in the neural network and $\mathcal{D}$ denote the training distribution. Then, the Hessian of the loss with respect to W is given by

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\frac{\partial^2 \mathcal{L}}{\partial W^2}\right] = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\frac{\partial f}{\partial W} \frac{\partial^2 \mathcal{L}}{\partial f^2} \frac{\partial f}{\partial W}^\top\right] + \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\frac{\partial \mathcal{L}}{\partial f} \frac{\partial^2 f}{\partial W^2}\right].$$

The first component, for standard losses like cross-entropy (CE) and mean squared error (MSE), is positive semi-definite and is generally known as the Gauss–Newton (GN) component (H_{GN}). Previous works have shown that this part closely tracks the overall Hessian during neural network training (Sankar et al., 2021), and thus most second-order methods approximate the GN component. Denoting $\frac{\partial \mathcal{L}(f(x),y)}{\partial W}$ by $G_{x,y} \in \mathbb{R}^{m\times n}$ and $g_{x,y} = \text{vec}(G_{x,y})$, for CE loss, it can also be shown that

$$H_{\text{GN}} = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\frac{\partial f}{\partial W} \frac{\partial^2 \mathcal{L}}{\partial f^2} \frac{\partial f}{\partial W}^\top\right] = \mathop{\mathbb{E}}_{\substack{x\sim\mathcal{D}_x \\ s\sim f(x)}}\left[g_{x,s} g_{x,s}^\top\right],$$

E Related work

The literature related to second order optimization within deep learning is very rich, with methods that can be broadly classified as Hessian-free and methods based on estimating the preconditioner H (which could refer to either H_{Ada} or H_{GN}). Hessian-free methods (Martens, 2010) generally tend to approximate the preconditioned step (for Newton's method) using Hessian vector products, but do not maintain an explicit form of the Hessian. Estimating H (Martens & Grosse, 2015a; Gupta et al., 2018a) methods maintain an explicit form of the preconditioner that could be efficiently stored as well as estimated.

E.1 Hessian-free

One of the seminal works related to second order optimization within deep learning was the introduction of Hessian-free optimization (Martens, 2010). The work demonstrated the effectiveness of using conjugate gradient (CG) for approximately solving the Newton step on multiple auto-encoder and classifications tasks. Multiple works (Martens & Sutskever, 2011; Cho et al., 2015) have extended this algorithm to other architectures such as recurrent networks and multidimensional neural nets. One of the recent works (Garcia et al., 2023) also takes motivation from this line of work, by approximately using single step CG for every update, along with maintaining a closed form for the inverse of the Hessian, for the single step to be effective.

E.2 Estimating Preconditioner

Given that it is costly to store the entire matrix H, various works have tried to estimate layer-wise H. KFAC (Martens & Grosse, 2015a) was one of the first work, that went beyond diagonal approximation and made a Kronecker product approximation to layer-wise H_{GN}. It showed that this structure approximately captures the per layer Hessian for MLPs. This approximation was extended to convolutional (Osawa et al., 2019) and recurrent (Martens et al., 2018) architectures. Subsequent works also improved the Hessian approximation, by further fixing the trace (Gao et al., 2021) as well as the diagonal estimates (George et al., 2018; Gao et al., 2020) of the approximation. A recent work (Eschenhagen et al., 2023) also demonstrated that K-FAC can be extended to large-scale training.

From the viewpoint of approximating Adagrad (Duchi et al., 2011b), Gupta et al. (2018a) introduced Shampoo, that also makes a Kronecker product approximation to H_{Ada}. One of the subsequent work (Ren & Goldfarb, 2021) introduced a modification of Shampoo, that was precisely estimating the layer-wise H_{GN} under certain distributional assumptions. Other works (Anil et al., 2021) introduced a distributed implementation of Shampoo, that has recently shown impressive performance for training large scale networks (Shi et al., 2023). Recently, another paper (Duvvuri et al., 2024) proposed a modification of Shampoo, empirically and theoretically demonstrating that the new estimator approximates H_{Ada} better than Shampoo's approximation. Our work shows that the square of Shampoo's approximation of H_{Ada} is nearly equivalent to the optimal Kronecker approximation.

F Comparison with extra square root in Adagrad based approaches

Multiple previous works (Balles et al., 2020; Lin et al., 2024) have tried to address the question of why Adagrad-based approaches like Adam and Shampoo, have an extra square root in their update compared to Hessian inverse in their updates. This question is primarily concerned with the final update to the weights being used in the optimization procedure, once we have approximated the Hessian.

The primary contribution of this work is completely orthogonal to this question. We are addressing the question of optimal Kronecker approximation of the Hessian, and its connection to Shampoo's Hessian approximation. This is orthogonal to the Hessian power used in the final update.