Purifying Shampoo: Investigating Shampoo's Heuristics by Decomposing its Preconditioner

Runa Eschenhagen[*,1] **Aaron Defazio**[2] **Tsung-Hsien Lee**[†]
Richard E. Turner[1,3] **Hao-Jun Michael Shi**[4]

[1]Department of Engineering, University of Cambridge
[2]Fundamental AI Research, Meta Superintelligence Labs, Meta Platforms, Inc.
[3]The Alan Turing Institute
[4]Infrastructure Optimizations, Meta Superintelligence Labs, Meta Platforms, Inc.

Abstract

The recent success of Shampoo in the AlgoPerf contest has sparked renewed interest in Kronecker-factorization-based optimization algorithms for training neural networks. Despite its success, Shampoo relies heavily on several heuristics such as learning rate grafting and stale preconditioning to achieve performance at-scale. These heuristics increase algorithmic complexity, necessitate further hyperparameter tuning, and lack theoretical justification. This paper investigates these heuristics from the angle of Frobenius norm approximation to full-matrix Adam and decouples the preconditioner's eigenvalues and eigenbasis updates. We show that grafting from Adam mitigates the staleness and mis-scaling of the preconditioner's *eigenvalues* and how correcting the eigenvalues directly eliminates the need for learning rate grafting. To manage the error induced by infrequent *eigenbasis* computations, we propose an adaptive criterion for determining the eigenbasis computation frequency motivated by terminating a warm-started QR algorithm. This criterion decouples the update frequency of different preconditioner matrices and enables us to investigate the impact of approximation error on convergence. These practical techniques offer a principled angle towards removing Shampoo's heuristics and developing improved Kronecker-factorization-based training algorithms.

1 Introduction

Structured non-diagonal, and especially Kronecker-factored, preconditioned stochastic gradient algorithms have been extensively studied for neural network training (Heskes, 2000; Martens, 2010; Martens & Grosse, 2015; Li, 2018; Gupta et al., 2018). Despite their promise, diagonally preconditioned methods like Adam have remained the de facto methods for training neural networks over the past decade (Duchi et al., 2011; Kingma & Ba, 2015). Recently, a distributed implementation of the Shampoo algorithm (Gupta et al., 2018; Anil et al., 2020; Shi et al., 2023) won the external tuning track of the AlgoPerf neural network training algorithm competition (Dahl et al., 2023; Kasimbeg et al., 2025).[1] This result has renewed interest in non-diagonally preconditioned training algorithms, inspiring methods like Muon (Jordan et al., 2024; Bernstein, 2025) and SOAP (Vyas et al., 2025a).

Correspondence to: re393@cam.ac.uk and hjmshi@meta.com.

[*]Work performed while an intern and external research collaborator at FAIR, Meta Platforms.

[†]Work performed while employed by AI and Systems Co-Design, Meta Platforms.

[1]The *external tuning* track requires a submission to specify a hyperparameter search space, in contrast to the hyperparameter-tuning-free *self-tuning* track.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

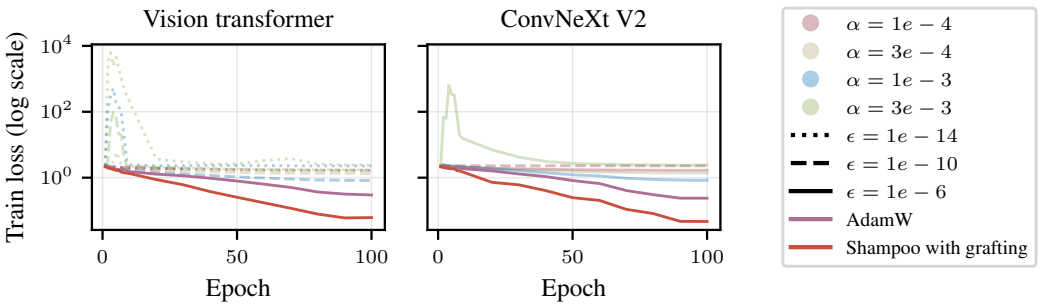

Figure 1: Shampoo with stale preconditioner (updating the root inverse matrices every $F = 100$ steps) without grafting for different choices of the learning rate α and ϵ on Imagewoof. All tested hyperparameter combinations underperform AdamW and, by extension, Shampoo with Adam grafting.

However, the winning Shampoo submission to the AlgoPerf competition relied on several crucial heuristics beyond Shampoo (Anil et al., 2020; Shi et al., 2023). Most notably, each layer's update is re-scaled by the update magnitude of a reference optimizer, a technique known as *learning rate grafting* (Agarwal et al., 2020); see Figure 1 for an illustration of its importance. Additionally, to reduce its computational overhead, the root-inverse of the preconditioner is only re-computed every 100 steps, resulting in the use of a *stale* preconditioner. Despite their empirical effectiveness, both heuristics lack theoretical justification and remain poorly understood.

In this paper, we investigate the role of these two heuristics by decoupling the updates of the preconditioner's eigenvalues and eigenbasis. In Section 3, we empirically demonstrate that Shampoo requires learning rate grafting in order to address the staleness and mis-scaling of the preconditioner's *eigenvalues*. Correcting Shampoo's eigenvalues at every step like in SOAP (Vyas et al., 2025a) removes the need for grafting. We further formalize this intuition by comparing bounds on the update magnitude of Shampoo and full-matrix Adam.

Given the importance of controlling the approximation error of the eigenvalues, we next consider the frequency of the *eigenbasis* updates in Section 4. Motivated by a termination criterion for the QR algorithm that bounds the relative error of the Kronecker factor approximation induced by the current eigenbasis (Golub & Van Loan, 2013), we propose an adaptive method for determining the eigenbasis update frequency. Our empirical results show that the approximation error of the Kronecker factors evolves during training and impacts convergence, depending on both the training stage and parameter's properties. Using an adaptive update frequency can improve Shampoo's training efficiency, especially when more frequent eigenbasis computations accelerate convergence.

2 Background

We consider neural network training as a standard stochastic optimization problem, where the goal is to minimize the expected loss function

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x}, \boldsymbol{y}) \sim p_{\mathcal{D}}(\boldsymbol{x}, \boldsymbol{y})} \big[\ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y}) \big], \tag{1}$$

where $f_{\boldsymbol{\theta}} : \mathbb{R}^N \to \mathbb{R}^c$ is the neural network prediction function with parameters $\boldsymbol{\theta} \in \mathbb{R}^d$ (flattened and concatenated into a vector), $p_{\mathcal{D}}(\boldsymbol{x}, \boldsymbol{y})$ is the joint data distribution from which inputs $\boldsymbol{x} \in \mathbb{R}^N$ and targets $\boldsymbol{y} \in \mathbb{R}^c$ are sampled, and $\ell : \mathbb{R}^c \times \mathbb{R}^c \to \mathbb{R}$ is the loss function. The neural network is commonly trained using preconditioned stochastic gradient methods that update the parameters at each iteration t by

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \boldsymbol{C}_t^{-p} \boldsymbol{g}_t = \boldsymbol{\theta}_t - \alpha_t \boldsymbol{Q}_{\boldsymbol{C}_t} \boldsymbol{\Lambda}_{\boldsymbol{C}_t}^{-p} \boldsymbol{Q}_{\boldsymbol{C}_t}^\intercal \boldsymbol{g}_t, \tag{2}$$

where $\boldsymbol{g}_t = \nabla_{\boldsymbol{\theta}_t} \ell(f_{\boldsymbol{\theta}_t}(\boldsymbol{x}_t), \boldsymbol{y}_t) \in \mathbb{R}^d$ is the stochastic (mini-batch) gradient with respect to the sample $(\boldsymbol{x}_t, \boldsymbol{y}_t) \sim p_{\mathcal{D}}(\boldsymbol{x}, \boldsymbol{y})$, $\alpha_t > 0$ is the step size, $p > 0$ is the exponent, and $\boldsymbol{C}_t \in \mathbb{R}^{d \times d}$ is a symmetric positive-definite *preconditioner* or *scaling* matrix. The second equality in Equation (2) expresses the update using the eigendecomposition of the preconditioner matrix $\boldsymbol{C}_t = \boldsymbol{Q}_{\boldsymbol{C}_t} \boldsymbol{\Lambda}_{\boldsymbol{C}_t} \boldsymbol{Q}_{\boldsymbol{C}_t}^\intercal$, where the eigenbasis matrix $\boldsymbol{Q}_t \in \mathbb{R}^{d \times d}$ is orthogonal and eigenvalue matrix $\boldsymbol{\Lambda}_{\boldsymbol{C}_t} \in \mathbb{R}^{d \times d}$ is diagonal.

We primarily focus on $\boldsymbol{C}_t$ as an accumulation of gradient outer products, i.e., $\bar{\boldsymbol{A}}_t = \sum_{s=1}^{t} \boldsymbol{g}_s \boldsymbol{g}_s^\intercal$ or $\boldsymbol{A}_t = \beta_2 \boldsymbol{A}_{t-1} + (1-\beta_2)\boldsymbol{g}_t \boldsymbol{g}_t^\intercal$ with $p = 1/2$, which correspond to full-matrix AdaGrad and RMSprop/Adam, respectively (Duchi et al., 2011; Kingma & Ba, 2015), although this can be generalized to other alternatives, like the natural gradient method (Amari, 1998); see Appendix A. When $\boldsymbol{C}_t$ is diagonal, this scaling recovers simplified versions of popular algorithms like AdaGrad (Duchi et al., 2011) or Adam (Kingma & Ba, 2015), ignoring additional modifications like exponential moving averages, momentum, bias corrections, etc. However, for non-diagonal $\boldsymbol{C}_t$, these methods require storing and possibly inverting dense $d \times d$ matrices, which is computationally prohibitive.

To address this, two structured approximations are commonly applied: (1) a layer-wise block-diagonal approximation, where each block captures pairwise correlations within each layer; (2) a Kronecker-factored approximation that takes a $mn \times mn$ matrix block for each layer and approximates it as a Kronecker product of two smaller matrices of size $m \times m$ and $n \times n$. The Kronecker product approximation is particularly convenient for computing the inverse-root matrix-vector product, leading to the design of algorithms like Kronecker-Factored Approximate Curvature (K-FAC) (Heskes, 2000; Martens & Grosse, 2015; Grosse & Martens, 2016; Martens et al., 2018; Eschenhagen et al., 2023), Shampoo (Gupta et al., 2018; Anil et al., 2020; Shi et al., 2023), and their variants (George et al., 2018; Gao et al., 2020; Ren & Goldfarb, 2021; Feinberg et al., 2023; Duvvuri et al., 2024; Lin et al., 2024b,a). Alternative approaches, such as Hessian-free or inexact Newton methods, avoid explicitly storing the preconditioner matrix, e.g., by leveraging Hessian-vector products (Martens, 2010; Li, 2018; Pooladzandi & Li, 2024).

2.1 Shampoo

The Shampoo preconditioner was originally developed as a Kronecker-factorized upper bound to full-matrix AdaGrad in its regret analysis (Gupta et al., 2018). For a weight matrix $\boldsymbol{W}_t \in \mathbb{R}^{m \times n}$ with stochastic gradient $\boldsymbol{G}_t \in \mathbb{R}^{m \times n}$ where $\boldsymbol{g}_t = \operatorname{vec}(\boldsymbol{G}_t)$, Shampoo stores symmetric positive semi-definite matrices that approximate an *idealized* preconditioner with $\boldsymbol{L}_t \approx \mathbb{E}[\boldsymbol{G}_t \boldsymbol{G}_t^\intercal] \in \mathbb{R}^{m \times m}$ and $\boldsymbol{R}_t \approx \mathbb{E}[\boldsymbol{G}_t^\intercal \boldsymbol{G}_t] \in \mathbb{R}^{n \times n}$, and updates the preconditioner and weight matrix by

$$\boldsymbol{L}_t = \beta_2 \boldsymbol{L}_{t-1} + (1-\beta_2)\boldsymbol{G}_t \boldsymbol{G}_t^\intercal, \tag{3}$$

$$\boldsymbol{R}_t = \beta_2 \boldsymbol{R}_{t-1} + (1-\beta_2)\boldsymbol{G}_t^\intercal \boldsymbol{G}_t, \tag{4}$$

$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \alpha_t \boldsymbol{L}_t^{-\frac{1}{4}} \boldsymbol{G}_t \boldsymbol{R}_t^{-\frac{1}{4}} \tag{5}$$

given $\beta_2 \in [0, 1)$.[2][3] The original Shampoo algorithm uses a sum to accumulate the gradient outer products, although the exponential moving average is more commonly used in practice. Therefore, we are interested in approximating the preconditioner of full-matrix Adam, given by $\boldsymbol{A}_t = \beta_2 \boldsymbol{A}_{t-1} + (1-\beta_2)\boldsymbol{g}_t \boldsymbol{g}_t^\intercal \approx \mathbb{E}[\boldsymbol{g}_t \boldsymbol{g}_t^\intercal]$.

The search direction or update in matrix form is given as $\boldsymbol{U}_t^{\text{Shampoo}} = -\boldsymbol{L}_t^{-\frac{1}{4}} \boldsymbol{G}_t \boldsymbol{R}^{-\frac{1}{4}}$ or, equivalently, $\boldsymbol{C}_t^{\text{Shampoo}} = (\boldsymbol{R}_t \otimes \boldsymbol{L}_t)^{\frac{1}{2}}$ with $p = 1/2$ in Equation (2). We also consider *Shampoo*^{2} defined as $\boldsymbol{C}_t^{\text{Shampoo}^2} = \boldsymbol{R}_t \otimes \boldsymbol{L}_t$, which is a tighter approximation to full-matrix AdaGrad (Morwani et al., 2025). Both updates can be generalized to higher-order tensors.

2.2 Eigenvalue-corrected Shampoo and SOAP

By leveraging a Kronecker product approximation, Shampoo's preconditioner is restricted to a Kronecker product structure for both its eigenvectors and eigenvalues. Specifically, if we have a preconditioner $\boldsymbol{C}_t = \boldsymbol{R}_t \otimes \boldsymbol{L}_t$ with eigendecompositions $\boldsymbol{L}_t = \boldsymbol{Q}_{\boldsymbol{L}_t} \boldsymbol{\Lambda}_{\boldsymbol{L}_t} \boldsymbol{Q}_{\boldsymbol{L}_t}^\intercal$ and $\boldsymbol{R}_t = \boldsymbol{Q}_{\boldsymbol{R}_t} \boldsymbol{\Lambda}_{\boldsymbol{R}_t} \boldsymbol{Q}_{\boldsymbol{R}_t}^\intercal$, then $\boldsymbol{C}_t$ has eigendecomposition $\boldsymbol{C}_t = (\boldsymbol{Q}_{\boldsymbol{R}_t} \otimes \boldsymbol{Q}_{\boldsymbol{L}_t})(\boldsymbol{\Lambda}_{\boldsymbol{R}_t} \otimes \boldsymbol{\Lambda}_{\boldsymbol{L}_t})(\boldsymbol{Q}_{\boldsymbol{R}_t} \otimes \boldsymbol{Q}_{\boldsymbol{L}_t})^\intercal$. Preconditioning $-\boldsymbol{g}_t$ with $\boldsymbol{C}_t^{\text{Shampoo}}$ and $p = 1/2$ in its matrix form is equivalent to the matrix transformation:

$$\boldsymbol{U}_t^{\text{Shampoo}} = -\boldsymbol{L}_t^{-\frac{1}{4}} \boldsymbol{G}_t \boldsymbol{R}_t^{-\frac{1}{4}} = -\boldsymbol{Q}_{\boldsymbol{L}_t} \boldsymbol{\Lambda}_{\boldsymbol{L}_t}^{-\frac{1}{4}} (\boldsymbol{Q}_{\boldsymbol{L}_t}^\intercal \boldsymbol{G}_t \boldsymbol{Q}_{\boldsymbol{R}_t}) \boldsymbol{\Lambda}_{\boldsymbol{R}_t}^{-\frac{1}{4}} \boldsymbol{Q}_{\boldsymbol{R}_t}^\intercal. \tag{6}$$

[2] We drop the subscript in the expectation, taken with respect to $(\boldsymbol{x}_t, \boldsymbol{y}_t) \sim p_{\mathcal{D}}(\boldsymbol{x}, \boldsymbol{y})$.

[3] A pseudo-inverse that ignores the null space of the matrix or perturbing the matrix by a regularization term $\epsilon \boldsymbol{I}$ for $\epsilon > 0$ can be used to handle the symmetric positive semi-definite case.

This can be interpreted as applying a Kronecker-factored coordinate-wise scaling to a gradient with changed basis, i.e., $\tilde{\boldsymbol{G}}_t = \boldsymbol{Q}_{\boldsymbol{L}_t}^\intercal \boldsymbol{G}_t \boldsymbol{Q}_{\boldsymbol{R}_t}$, scaling the transformed gradient, and converting it back to its original basis.

Instead of restricting the eigenvalues to be a Kronecker product, Liu et al. (2018) and George et al. (2018) have proposed *correcting* the eigenvalues by decoupling the scaling from the basis in K-FAC. This involves computing a separate scaling matrix $\boldsymbol{D}_t \approx \mathbb{E}\big[\tilde{\boldsymbol{G}}_t^{\odot 2}\big] \in \mathbb{R}^{m \times n}$ and using it in place of the preconditioner's original eigenvalues $\operatorname{mat}\operatorname{diag}(\boldsymbol{\Lambda}_{\boldsymbol{R}_t} \otimes \boldsymbol{\Lambda}_{\boldsymbol{L}_t})$.[4][5]

Anil et al. (2020) noted that this correction can also be applied to Shampoo. Most recently, Vyas et al. (2025a) presented promising empirical results for language models using an instance of eigenvalue-corrected Shampoo called SOAP, which updates the scaling by $\boldsymbol{D}_t = \beta_2 \boldsymbol{D}_{t-1} + (1-\beta_2)\tilde{\boldsymbol{G}}_t^{\odot 2}$. Since then, SOAP has also been shown to perform well for physics-informed neural networks (Wang et al., 2025) and to reduce outlier features in transformers, which potentially improves quantization (He et al., 2024). We refer to our practical instantiation of eigenvalue-corrected Shampoo as *EShampoo* (Appendix B, Algorithm 2), which uses the same $\boldsymbol{D}_t$ as SOAP. See Appendix D.1 for clarification on the distinction between eigenvalue correction, EShampoo, and SOAP.

3 Learning rate grafting compensates for Shampoo's eigenvalues

Originally motivated by decoupling an optimizer's update magnitude from its direction to account for different implicit learning rate schedules, Agarwal et al. (2020) proposed *learning rate grafting*, a technique that combines the layer-wise update of one optimizer with the layer-wise update magnitude of another. For Shampoo, this means taking Shampoo's layer-wise update $\boldsymbol{U}_t^{\text{Shampoo}}$ and rescaling it by the Frobenius norm of the grafting method's update $\boldsymbol{U}_t^{\text{Grafting}}$ (typically Adam):

$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t + \alpha_t \frac{||\boldsymbol{U}_t^{\text{Grafting}}||_F}{||\boldsymbol{U}_t^{\text{Shampoo}}||_F} \boldsymbol{U}_t^{\text{Shampoo}}. \tag{7}$$

A complete description of Shampoo with Adam grafting is given in Appendix B, Algorithm 3.

This approach has been critical to Shampoo's empirical success (Anil et al., 2020; Shi et al., 2023; Kasimbeg et al., 2025). As shown in Figure 1, Shampoo without grafting (and updating the root inverse matrices every $F = 100$ steps) underperforms AdamW, with many hyperparameter settings diverging. Anil et al. (2020) suggests that grafting is used to account for differences in magnitude of the eigenspectrum of the Kronecker factors for different layers, as well as the infrequent updates of the Kronecker factors and their inverse roots (see Anil et al. (2020), Appendix G). However, these claims have not been thoroughly investigated.

To focus our empirical investigation, we study Shampoo with Adam grafting and $F = 100$, which was the winning configuration that was used in the external tuning track of the AlgoPerf competition (Kasimbeg et al., 2025). Since grafting re-scales the layer-wise update based on its magnitude, we analyze the Frobenius norm of the updates of full-matrix and diagonal Adam, Shampoo, and EShampoo. The magnitude of the eigendecomposed update in Equation (2) with Kronecker-factored $\boldsymbol{Q}_{\boldsymbol{C}_t} = \boldsymbol{Q}_{\boldsymbol{R}_t} \otimes \boldsymbol{Q}_{\boldsymbol{L}_t}$ is determined by the norm of the stochastic gradient $\boldsymbol{G}_t$ and the eigenvalues of $\boldsymbol{C}_t$:

Lemma 1. *Let $\boldsymbol{U} = \boldsymbol{Q}_{\boldsymbol{L}}(\boldsymbol{D}^{\odot -p} \odot (\boldsymbol{Q}_{\boldsymbol{L}}^\intercal \boldsymbol{G} \boldsymbol{Q}_{\boldsymbol{R}}))\boldsymbol{Q}_{\boldsymbol{R}}^\intercal \in \mathbb{R}^{m \times n}$ be the generalized eigendecomposed Kronecker-factored update given by orthogonal matrices $\boldsymbol{Q}_{\boldsymbol{L}} \in \mathbb{R}^{m \times m}$, $\boldsymbol{Q}_{\boldsymbol{R}} \in \mathbb{R}^{n \times n}$, and dense scaling matrix $\boldsymbol{D} \in \mathbb{R}^{m \times n}$, with $p > 0$. Then we have:*

$$(\max_{i,j} \boldsymbol{D}_{i,j})^{-p} ||\boldsymbol{G}||_F \leq ||\boldsymbol{U}||_F \leq (\min_{i,j} \boldsymbol{D}_{i,j})^{-p} ||\boldsymbol{G}||_F. \tag{8}$$

Lemma 1 covers multiple algorithms. We can recover the idealized Adam update by setting $\boldsymbol{Q}_{\boldsymbol{L}} = \boldsymbol{I} \in \mathbb{R}^{m \times m}$, $\boldsymbol{Q}_{\boldsymbol{R}} = \boldsymbol{I} \in \mathbb{R}^{n \times n}$, and $\boldsymbol{D} = \mathbb{E}\big[\boldsymbol{G}^{\odot 2}\big]$ with $p = 1/2$. The idealized Shampoo update can be recovered through the choice of $\boldsymbol{D} = \operatorname{mat}\operatorname{diag}(\boldsymbol{\Lambda}_{\boldsymbol{R}} \otimes \boldsymbol{\Lambda}_{\boldsymbol{L}})$ with $p = 1/4$, $\boldsymbol{L} = \mathbb{E}[\boldsymbol{G}\boldsymbol{G}^\intercal] = \boldsymbol{Q}_{\boldsymbol{L}} \boldsymbol{\Lambda}_{\boldsymbol{L}} \boldsymbol{Q}_{\boldsymbol{L}}^\intercal$ and $\boldsymbol{R} = \mathbb{E}[\boldsymbol{G}^\intercal \boldsymbol{G}] = \boldsymbol{Q}_{\boldsymbol{R}} \boldsymbol{\Lambda}_{\boldsymbol{R}} \boldsymbol{Q}_{\boldsymbol{R}}^\intercal$ (or $p = 1/2$ for Shampoo^2). Idealized EShampoo is recovered with the choice of $\boldsymbol{D} = \mathbb{E}\big[(\boldsymbol{Q}_L^\intercal \boldsymbol{G} \boldsymbol{Q}_R)^{\odot 2}\big]$ and $p = 1/2$ instead.

[4] $\boldsymbol{X}^{\odot 2}$ denotes the element-wise square.

[5] $\operatorname{mat}\operatorname{diag}(\cdot)$ takes a $mn \times mn$ matrix and reshapes its diagonal into a $m \times n$ matrix.

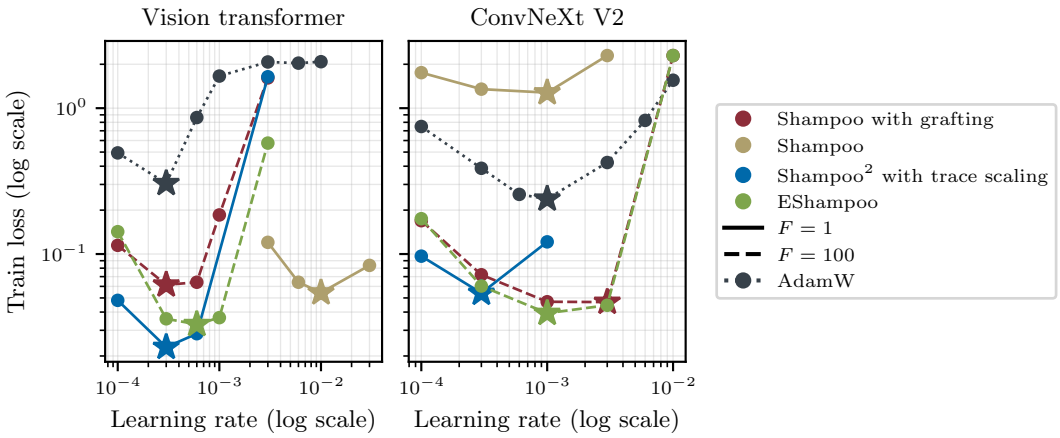

Figure 2: Training results with different Shampoo variants and eigendecomposition frequencies F on the Imagewoof dataset. Shampoo with eigenvalue correction achieves a better training loss compared to Shampoo with Adam grafting, and the optimal learning rate for Adam transfers to both variants.

Proposition 1. *Assume that $\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal]$ is symmetric positive definite. The magnitude of the idealized updates for full-matrix Adam, diagonal Adam, and EShampoo are all bounded by the power of the extreme eigenvalues of full-matrix Adam:*

$$\lambda_{\max}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal])^{-p}\|\boldsymbol{G}\|_F \leq \|\boldsymbol{U}\|_F \leq \lambda_{\min}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal])^{-p}\|\boldsymbol{G}\|_F, \tag{9}$$

for all $p > 0$. However, under the simplifying assumption that $\mathbb{E}[\boldsymbol{G}] = \boldsymbol{0}$ and $\boldsymbol{G}_{i,j}$ is independent from $\boldsymbol{G}_{k,l}$ for $(i,j) \neq (k,l)$ and has bounded second moment, $\lambda_{\min}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal]) \leq \mathbb{E}[\boldsymbol{G}_{i,j}^2] \leq \lambda_{\max}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal])$ and Shampoo has dimension-dependent bounds:

$$m^{-p/2}n^{-p/2}\lambda_{\max}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal])^{-p}\|\boldsymbol{G}\|_F \leq \|\boldsymbol{U}\|_F \leq m^{-p/2}n^{-p/2}\lambda_{\min}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal])^{-p}\|\boldsymbol{G}\|_F. \tag{10}$$

See Appendix C for the proofs of Lemma 1 and Proposition 1.

This highlights a key issue: Shampoo's update magnitude can be mis-scaled relative to Adam and EShampoo, especially due to dimension-dependent factors. The basis does not influence the update magnitude – only the eigenvalues do. While this additional scaling can be absorbed into the learning rate when only handling a single matrix, it is potentially problematic when one needs to handle multiple parameters simultaneously. In addition, the use of stale eigenvalues can result in update magnitudes that lie outside of the bounds of full-matrix Adam. This leads us to the following hypothesis:

The role of learning rate grafting in Shampoo

Learning rate grafting compensates for the scaling and staleness of Shampoo's eigenvalues.

From the Frobenius norm approximation perspective, using Shampoo^2 (via $\boldsymbol{C}_t^{\text{Shampoo}^2}$) yields a tighter approximation to full-matrix Adam compared to $\boldsymbol{C}_t^{\text{Shampoo}}$ (Morwani et al., 2025), which addresses the mismatch of the eigenvalues' exponent in Equation (10). Additionally, rescaling the preconditioner by $S^{-1} = \operatorname{Tr}(\boldsymbol{R}_t)^{-1} = \operatorname{Tr}(\boldsymbol{L}_t)^{-1}$ ensures exactness when full-matrix Adam $\boldsymbol{A}_t$ is a Kronecker product (Morwani et al., 2025). The scaling S^{-1} has also been previously introduced for Tensor Normal Training (Ren & Goldfarb, 2021, TNT) to approximate the Fisher information matrix.

Based on our observations, we can make several predictions:

1. With $F = 1$, Shampoo without grafting should perform well when layer scalings are similar, but may struggle with highly variable parameter shapes.
2. Using $S^{-1}\boldsymbol{C}_t^{\text{Shampoo}^2}$ with $F = 1$ should address scaling and staleness and match Shampoo with grafting. This is exact when full-matrix Adam decomposes into a Kronecker product.
3. Updating an eigenvalue correction at every iteration (e.g. like in SOAP) should also address scaling and staleness, matching Shampoo with grafting.

Empirical validation. We ablate different variants of Shampoo on the Imagewoof dataset with vision transformer (ViT) and ConvNeXt V2 models. We plot the final training loss after 100 epochs against the learning rate. Following the specification of the AlgoPerf Shampoo submission, we update the preconditioner every 100 steps when grafting the learning rate from Adam, and the eigenbasis when using eigenvalue correction. Our results are presented in Figure 2. We observe that all three predictions are confirmed: Shampoo^2 scaled by S^{-1} or eigenvalue correction is able to match or even surpass Shampoo with grafting on both problems, and Shampoo without grafting can match the final loss for the Imagewoof ViT workload, but not for the ConvNeXt V2 model.

Hyperparameter transfer. The optimal learning rate of Adam transfers to Shampoo with grafting and EShampoo. However, it is not apparent whether the optimal learning rate transfers to Shampoo^2 scaled by S^{-1}. Learning rate transfer is only sensible when jointly transferring ϵ since it affects the effective step size, but Shampoo does not directly add ϵ to the root of the eigenvalues, as done in Adam and other diagonal-scaling-based optimizers. While we have not tested this, the optimal learning rate may potentially transfer when matching the effective ϵ.

Computational and memory costs. While updating Shampoo every iteration incurs a prohibitive computational overhead, the eigenvalue correction is only slightly more expensive than Adam grafting since it requires more matrix products. EShampoo adds the same memory overhead to Shampoo as Adam grafting, specifically requiring an additional d-dimensional buffer for the second moment accumulation. One could reduce this overhead by adopting techniques such as Adam-mini, which only requires a single scalar per parameter block, defined according to the Hessian structure (Zhang et al., 2025b), and could be applied to both grafting and the eigenvalue correction. Alternatively, Liu et al. (2025) propose to scale Muon to approximately match the RMS norm of Adam's update to enable the re-use of Adam's hyperparameters, which may provide a practical heuristic that removes grafting without any memory overhead. Finally, an eigenvalue correction for the individual Kronecker factors can also correct for the staleness of their eigenvalues and has, in fact, been shown to also remove the need for grafting, while reducing the necessary buffer size from mn to $m + n$ for each $m \times n$ weight matrix (Lin et al., 2025).

Applicability to other methods. The issue of stale eigenvalues extends to all sparsely-updated Kronecker-factored preconditioners such as K-FAC variants and TNT. Although we are unaware of any work that combines K-FAC or TNT with grafting, it is common practice to apply the Adam update to gradients preconditioned with K-FAC (Pauloski et al., 2021; Osawa et al., 2023a,b; Eschenhagen et al., 2023), which we hypothesize could serve a similar function as grafting.

4 Controlling the approximation error induced by stale eigenbases

While frequent updates to the preconditioner's eigenvalues are important, the impact of periodically computing the eigenbasis remains less clear. The optimal frequency of the eigenbasis computation should be chosen to balance the time of each iteration against the method's per-iteration convergence rate, and may depend on the stage of training, layer type, and the specific Kronecker factors ($\boldsymbol{L}_t$, $\boldsymbol{R}_t$) involved. Shampoo currently uses a fixed update frequency for all matrices, which ignores these distinctions and the computational cost associated with different parameter shapes.

4.1 Can we adaptively control the approximation error induced by the stale eigenbasis?

In SOAP, the initial eigenbasis is computed via an eigendecomposition during the first iteration, then is refined by a single power iteration and QR decomposition at all subsequent iterations, c.f. Algorithm 4 in Vyas et al. (2025a), attributed to Wang et al. (2024).[6] The method corresponds to a single iteration of a warm-started simultaneous iteration, an extension of the power iteration for iteratively computing eigendecompositions (Golub & Van Loan, 2013). However, this approach may still yield a poor approximation to the true eigenbasis, especially when the eigenbasis changes rapidly due to the choice of β_2 or the dynamics of the stochastic gradient.

To address this, we propose controlling the error induced by the eigenbasis using a *warm-started QR algorithm* that iterates until a relative approximation error criterion is satisfied. This approach offers three major benefits: (1) each Kronecker factor can be treated independently, allowing adaptive frequency across different layers, factors, and stages of training; (2) evaluating the criterion with the

[6] This approach was referred to as randomized SVD in Wang et al. (2024), Appendix B.

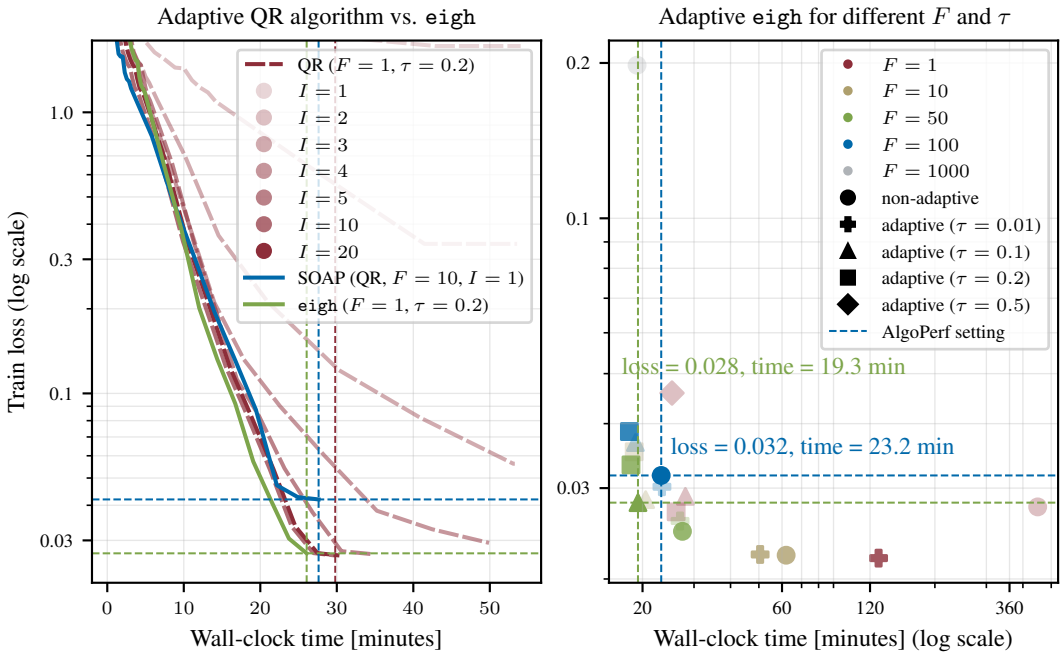

Figure 3: All configurations are for EShampoo. (**left**) On the Imagewoof ViT problem, setting the maximum number of iterations $I < 10$ with threshold $\tau = 0.2$ for the adaptive QR algorithm leads to significant increase in wall-clock time compared to using adaptive `eigh`. Even with $I = 10$, adaptive `eigh` is faster. The default SOAP setting achieves worse final loss and is also slightly slower. (**right**) Using the adaptive criterion to determine when to skip the eigendecomposition (`eigh`) improves efficiency by 20% in wall-clock time compared to updating every 100 iterations (AlgoPerf setting).

last computed eigenbasis prior to the first QR iteration enables us to *skip* eigenbasis updates; and (3) the error in the eigenbasis can be controlled through a threshold $\tau \in [0, 1)$, quantifying *inexactness*.

Consider a single Kronecker factor $\boldsymbol{L}_t \in \mathbb{R}^{m \times m}$ without loss of generality. We want to solve inexactly for an orthogonal matrix $\boldsymbol{Q}_{\boldsymbol{L}_t}$ such that we obtain an approximation to the eigendecomposition $\boldsymbol{L}_t = \boldsymbol{Q}_{\boldsymbol{L}_t} \boldsymbol{\Lambda}_{\boldsymbol{L}_t} \boldsymbol{Q}_{\boldsymbol{L}_t}^\intercal$, with the diagonal eigenvalues matrix $\boldsymbol{\Lambda}_{\boldsymbol{L}_t}$. If the previous approximate eigenbasis $\hat{\boldsymbol{Q}}_{\boldsymbol{L}_{t-1}}$ is a good approximation to the current eigenbasis $\boldsymbol{Q}_{\boldsymbol{L}_t}$, then the approximate eigenvalues induced by the stale eigenbasis $\hat{\boldsymbol{\Lambda}}_{\boldsymbol{L}_t} = \hat{\boldsymbol{Q}}_{\boldsymbol{L}_{t-1}}^\intercal \boldsymbol{L}_t \hat{\boldsymbol{Q}}_{\boldsymbol{L}_{t-1}}$ should be nearly diagonal.

We can leverage this observation to define a unified criterion for skipping the eigenbasis computation and terminating the warm-started QR algorithm. Diagonalizing $\hat{\boldsymbol{\Lambda}}_{\boldsymbol{L}_t}$ gives an approximation of $\boldsymbol{L}_t$, specifically, $\hat{\boldsymbol{L}}_t = \hat{\boldsymbol{Q}}_{\boldsymbol{L}_{t-1}} \operatorname{diag}(\hat{\boldsymbol{\Lambda}}_{\boldsymbol{L}_t}) \hat{\boldsymbol{Q}}_{\boldsymbol{L}_{t-1}}^\intercal$.[7] We can bound the relative error of this approximation induced by the stale eigenbasis, or equivalently the relative error in the Frobenius norm of the off-diagonal entries of $\hat{\boldsymbol{\Lambda}}_{\boldsymbol{L}_t}$, which is cheaper to compute:

Adaptive eigenbasis computation frequency criterion

$$\frac{||\boldsymbol{L}_t - \hat{\boldsymbol{L}}_t||_F}{||\boldsymbol{L}_t||_F} = \frac{||\hat{\boldsymbol{Q}}_{\boldsymbol{L}_{t-1}}^\intercal \boldsymbol{L}_t \hat{\boldsymbol{Q}}_{\boldsymbol{L}_{t-1}} - \operatorname{diag}(\hat{\boldsymbol{\Lambda}}_{\boldsymbol{L}_t})||_F}{||\hat{\boldsymbol{Q}}_{\boldsymbol{L}_{t-1}}^\intercal \boldsymbol{L}_t \hat{\boldsymbol{Q}}_{\boldsymbol{L}_{t-1}}||_F} = \frac{||\hat{\boldsymbol{\Lambda}}_{\boldsymbol{L}_t} - \operatorname{diag}(\hat{\boldsymbol{\Lambda}}_{\boldsymbol{L}_t})||_F}{||\hat{\boldsymbol{\Lambda}}_{\boldsymbol{L}_t}||_F} \leq \tau. \quad (11)$$

This condition provides a guarantee of the quality of the eigendecomposition approximation. If the condition is satisfied, we reuse the previous eigenbasis $\hat{\boldsymbol{Q}}_{\boldsymbol{L}_t} = \hat{\boldsymbol{Q}}_{\boldsymbol{L}_{t-1}}$; otherwise, we refine it through the QR algorithm until the criterion is met. A complete pseudocode is provided in Appendix B, Algorithm 4. Note that evaluating the approximate eigenvalues $\hat{\boldsymbol{\Lambda}}_{\boldsymbol{L}_t}$ at the first step requires two matrix multiplications, which is significantly cheaper than computing a step of the QR algorithm.

[7] $\operatorname{diag}(\cdot)$ takes a vector/matrix and returns a diagonal matrix with the vector/matrix's diagonal on its diagonal.

Table 1: Results on a subset of the AlgoPerf workloads. We show the mean and standard error of the steps/time to the targets across the runs that reach them. See Appendix E, Table 3 for more results.

Workload	Shampoo Variant	Hits Target	Steps	Time [min]
FastMRI	Adam grafting ($F = 100$)	$4/5$	4301 ± 109	13.96 ± 0.44
	$\boldsymbol{C}^{\text{EShampoo}}$ ($F = 100$)	$5/5$	2536 ± 66	10.44 ± 0.21
	$\boldsymbol{C}^{\text{EShampoo}}$ ($\tau = 0.1$, $F = 50$)	$5/5$	2468 ± 145	10.81 ± 0.72
ImageNet ViT	Adam grafting ($F = 100$)	$1/1$	79907	894.27
	$\boldsymbol{C}^{\text{EShampoo}}$ ($F = 100$)	$1/1$	76226	894.85
	$\boldsymbol{C}^{\text{EShampoo}}$ ($\tau = 0.1$, $F = 50$)	$1/1$	77459	935.89
OGBG	Adam grafting ($F = 100$)	$2/5$	12574 ± 708	39.20 ± 1.88
	$\boldsymbol{C}^{\text{EShampoo}}$ ($F = 100$)	$3/5$	8320 ± 1203	33.02 ± 4.05
	$\boldsymbol{C}^{\text{EShampoo}}$ ($\tau = 0.1$, $F = 50$)	$5/5$	7117 ± 328	27.55 ± 3.49

4.2 Does this adaptivity translate to efficiency gains?

The practical efficiency gains of adaptively controlling the approximation error depend on multiple factors. First, the efficiency of the QR algorithm is determined by how its computational cost compares to a standard eigendecomposition, which in turn depends on both the cost of each QR iteration and the total number of iterations needed to satisfy the threshold τ. Second, evaluating the criterion adds a small constant overhead independent of the actual frequency of eigendecompositions.

Interestingly, we found that setting a smaller maximum number of QR iterations ($I < 10$) per step results in a significant slowdown in wall-clock time, as the total number of QR iterations accumulated over all training steps is higher than when using a larger maximum ($I \geq 10$). However, even with a larger number of maximum iterations, the QR algorithm was slightly more expensive than computing eigendecompositions with `torch.linalg.eigh` (shortened as `eigh`) whenever Equation (11) does not hold. The default configuration of SOAP, which computes a single step of the warm-started simultaneous iteration every 10 steps, was also slightly slower in wall-clock time compared to adaptively computing eigendecompositions and reaches a worse final loss. This result conflicts with the observation in Figure 7 in Vyas et al. (2025a), which may be due to differences in the types of workloads considered; see Figure 3 (left).

Because of this result, we primarily leverage the criterion for determining the frequency of calling `eigh`. To further bound the total possible number of eigendecompositions performed over the course of training, we evaluate the criterion at a fixed frequency F as opposed to every step. We ablate different choices of F and τ in Figure 3 (right). We compare against the baseline setting of the winning AlgoPerf submission, which re-computes the eigendecomposition of all factor matrices every 100 steps on the Imagewoof ViT problem. We observe that this setting without adaptivity requires 20% more wall-clock time compared to an adaptive setting ($\tau = 0.1$, $F = 50$).

To test whether the results generalize to other problems, we consider a subset of the AlgoPerf workloads and compare the winning Shampoo submission with Adam grafting to: (1) EShampoo with the same fixed eigenbasis computation frequency of $F = 100$; and (2) EShampoo with $\tau = 0.1$ and $F = 50$; see Table 1. First, as predicted by Section 3, EShampoo with the fixed eigenbasis update frequency matches or outperforms Shampoo with Adam grafting in steps and wall-clock time, using the same hyperparameters. Second, using the adaptive criterion with $\tau = 0.1$ and $F = 50$ matches the fixed frequency $F = 100$ for the FastMRI and OGBG workloads, and does slightly worse for ImageNet ViT. We also present results with other fixed and adaptive frequencies in Appendix E, Table 3. Overall, we find that the performance on the considered problems is quite robust to the eigenbasis computation frequency (c.f. Appendix E, Table 3). We expect that when Shampoo's convergence benefits from more frequent eigenbasis updates, adaptively determining the frequency can result in higher efficiency.

4.3 What patterns of adaptivity emerge?

The criterion in Equation (11) also provides insight into how rapidly the eigenbasis changes for different parameter types. On Imagewoof ViT, we set $\tau = 0.1$ and check the criterion at every

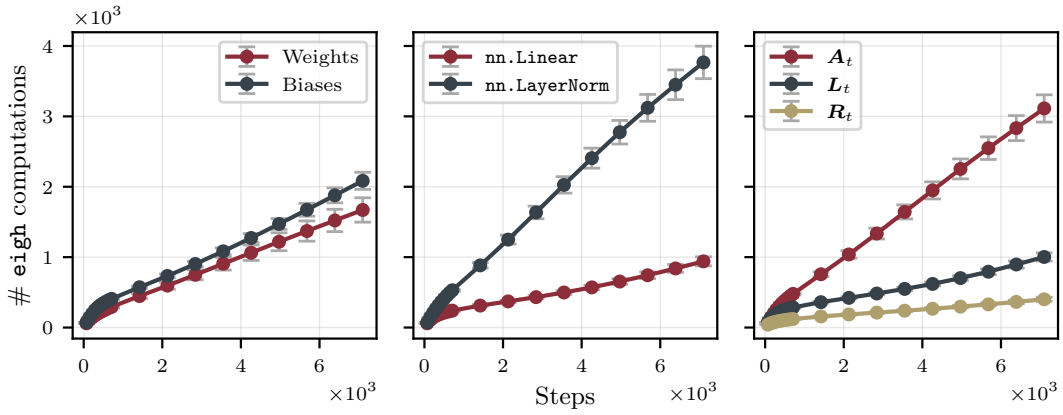

Figure 4: We show the mean with standard error across preconditioners corresponding to the labels in the legends, for EShampoo with $\tau = 0.01$ and $F = 1$ on Imagewoof ViT. The eigenbases for biases and layer normalization parameters are changing faster than for weight matrices and linear layers, respectively.

step ($F = 1$); see Figure 4. We observe more frequent eigendecomposition computations early in training that trend towards a constant update frequency at the end of training. When comparing different parameter types, we find that the eigenbases of preconditioners for bias terms evolve more rapidly than those for weights. Similarly, layer normalization parameters require almost $4\times$ as many eigenbasis updates as linear layer parameters.

We also compare the number of eigendecompositions across the two Kronecker factors $\boldsymbol{L}_t$ and $\boldsymbol{R}_t$ for weight matrices, and $\boldsymbol{A}_t$ for biases and layer normalization parameters.[8] The eigenbases of $\boldsymbol{A}_t$ are updated most frequently, followed by $\boldsymbol{L}_t$, then $\boldsymbol{R}_t$. The same trend holds when removing learnable layer norm parameters and for ConvNeXt V2 trained on the same dataset (c.f. Appendix E, Figure 7). A similar analysis for a Llama 3 model with 324 million parameters trained on 3.2 billion tokens of C4 data reveals a different trend, namely, $\boldsymbol{L}_t$ evolves faster than $\boldsymbol{R}_t$ and $\boldsymbol{A}_t$, which correspond to RMS normalization parameters (c.f. Appendix E, Figure 8).

4.4 How does the error induced by the eigenbasis affect convergence?

While we have focused on controlling the error induced by the preconditioner's stale eigenbasis, our primary concern is its effect on convergence, rather than the error itself. In the Imagewoof ViT setting, we find that the approximation quality (determined by τ) during early iterations is far more critical than during later iterations; see Figure 5 (left). Specifically, setting $\tau = 0.8$ for the first 90% and $\tau = 0.01$ for the last 10% of iterations yields a final loss nearly identical to using $\tau = 0.8$ throughout all of training ($\approx 0.09 - 0.1$). In contrast, setting $\tau = 0.01$ for the first 10% and $\tau = 0.8$ for the remaining 90% of iterations leads to a significantly better final loss (≈ 0.04).

Remarkably, freezing the eigenbasis after the first iteration (by setting $\tau = 0.99$ for all of training) still significantly outperforms AdamW (≈ 0.12 vs. 0.3).[9] This highlights that frequent eigenbasis computations are most beneficial early in training, consistent with previous observations on Shampoo, PSGD, and SOAP (Ishikawa & Yokota, 2024; Walters et al., 2025; Nestler, 2025; Vyas et al., 2025b).

To further investigate the discrepancy in the rate of change in the eigenbases corresponding to 1D and 2D parameters (c.f. Figure 4, right), we compare an Imagewoof ViT run with $\tau = 0.01$ for 2D parameters and no change of basis, i.e. Adam, for 1D parameters and vice versa. Surprisingly, using Adam for 1D parameters does not affect Shampoo's convergence or final loss. Conversely, running Adam for 2D parameters and changing the basis according to $\tau = 0.01$ for 1D parameters closely matches Adam's convergence and final loss; see Figure 5 (right).

[8] Full-matrix Adam is also used for weight matrices $\boldsymbol{W}_t \in \mathbb{R}^{m \times n}$ with $mn \leq$ `max_preconditioner_dim`, which is a hyperparameter to control the largest possible dimension of the preconditioner.

[9] For 5 out of 172 Kronecker factors, the eigenbasis is computed twice instead of just once.

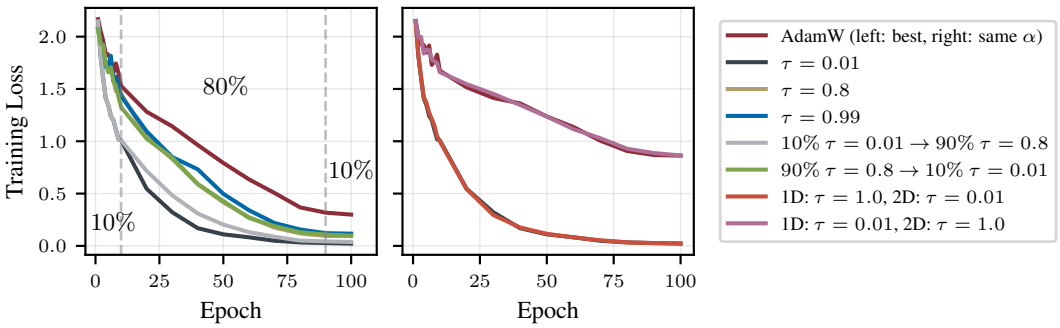

Figure 5: All configurations are for EShampoo. (**left**) The error in the eigenbases is dramatically more important for early iterations. A single eigenbasis computation at the first iteration ($\tau = 0.99$) is sufficient to outperform AdamW. (**right**) The difference between the convergence behavior of AdamW and EShampoo on this problem can be exclusively attributed to the eigenbases corresponding to 2D parameters.

Given that 45% of the preconditioner matrices correspond to 1D parameters and their eigenbases change more rapidly, this suggests that many eigendecompositions are not needed at all. For example, using $\tau = 0.01$ for all parameters increases runtime by $2.9\times$ compared to only applying EShampoo to 2D parameters and Adam to 1D parameters, with no improvement in final loss. In practice, SOAP already uses Adam for 1D parameters in order to reduce its computational and memory overhead (Vyas et al., 2025a).

5 Discussion and conclusion

In this paper, we demonstrate that frequently updating the eigenvalues while periodically updating the eigenbasis of Shampoo's preconditioner provides a principled and practical approach for eliminating grafting. It remains an open question whether the same correction applies directly to the AdaGrad summation, or if a more sophisticated, basis-aware eigenvalue correction is needed (c.f. Appendix D.2). We also show how controlling the approximation error induced by the stale eigenbasis can improve efficiency. In order to determine the eigenbasis computation frequency in a truly problem-agnostic manner, we must understand how approximation error impacts convergence, as well as integrate systems-level considerations, such as batched kernel efficiency, into our algorithmic design. Further exploration is needed to understand how these trade-offs and techniques scale to larger models, such as large language models. While our work focuses on Shampoo, the ideas are not limited to the AdaGrad family, and can be adapted to other methods such as K-FAC and TNT.

From a theoretical perspective, a major open question is how to incorporate approximation quality into regret bounds for Shampoo (c.f. Appendix D.1). Finally, while we have implicitly treated full-matrix Adam as the right algorithm to approximate, alternative interpretations of Shampoo may better explain Shampoo's effectiveness in practice (Carlson et al., 2015a,b; Benzing, 2022; Bernstein & Newhouse, 2024; Maes et al., 2024; Pethick et al., 2025; Zhang et al., 2025a; Xie et al., 2025).

Acknowledgments

We thank Anna Cai, Parameswaran Raman, and Ke Sang for their in-depth review of the paper. We are grateful for insightful discussions with Ganesh Ajjanagadde, Rohan Anil, Anna Cai, Rong Jin, Jihao Andreas Lin, Bruno Mlodozeniec, Vinay Rao, Isaac Reid, and Xingyu (Alice) Yang. We also acknowledge managerial support from Yuchen Hao, Guna Lakshminarayanan, Maxim Naumov, Sandeep Parab, Chunqiang Tang, and Lin Xiao. Lastly, we thank the anonymous reviewers for their productive feedback and suggestions of experiments and Kyunghun Nam and Yushun Zhang for pointing out minor errors in a preprint of this work.

Runa Eschenhagen is supported by ARM, the Cambridge Trust, and the Qualcomm Innovation Fellowship. Richard E. Turner is supported by Google, Amazon, ARM, Improbable and an EPSRC Prosperity Partnership (EP/T005386/1) and the EPSRC Probabilistic AI Hub (EP/Y028783/1).

References

Naman Agarwal, Rohan Anil, Elad Hazan, Tomer Koren, and Cyril Zhang. Disentangling adaptive gradient methods from learning rates. arXiv2002.11803, 2020.

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2), 1998.

Kang An, Yuxing Liu, Rui Pan, Shiqian Ma, Donald Goldfarb, and Tong Zhang. ASGO: Adaptive structured gradient optimization. arXiv 2503.20762, 2025.

Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. Scalable second order optimization for deep learning. arXiv 2002.09018, 2020.

Frederik Benzing. Gradient descent on neurons and its link to approximate second-order optimization. In *ICML*, 2022.

Albert S Berahas, Majid Jahani, Peter Richtárik, and Martin Takáč. Quasi-Newton methods for deep learning: Forget the past, just sample. arXiv 1901.09997, 2020.

Alberto Bernacchia, Mate Lengyel, and Guillaume Hennequin. Exact natural gradient in deep linear networks and its application to the nonlinear case. In *NeurIPS*, 2018.

Jeremy Bernstein. Deriving Muon, 2025. URL https://jeremybernste.in/writing/deriving-muon.

Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology. arXiv 2409.20325, 2024.

Lucas Beyer, Xiaohua Zhai, and Alexander Kolesnikov. Better plain ViT baselines for ImageNet-1k. arXiv 2205.01580, 2022.

Raghu Bollapragada, Jorge Nocedal, Dheevatsa Mudigere, Hao-Jun Shi, and Ping Tak Peter Tang. A progressive batching L-BFGS method for machine learning. In *International Conference on Machine Learning*, pp. 620–629. PMLR, 2018.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM review*, 60(2):223–311, 2018.

David Carlson, Volkan Cevher, and Lawrence Carin. Stochastic spectral descent for restricted Boltzmann machines. In *ICML*, 2015a.

David Edwin Carlson, Edo Collins, Ya-Ping Hsieh, Lawrence Carin, and Volkan Cevher. Preconditioned spectral descent for deep learning. In *NIPS*, 2015b.

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. arXiv 2306.07179, 2023.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12(61), 2011.

Sai Surya Duvvuri, Fnu Devvrit, Rohan Anil, Cho-Jui Hsieh, and Inderjit S Dhillon. CASPR: Combining axes preconditioners through Kronecker approximation for deep learning. In *ICLR*, 2024.

Runa Eschenhagen, Alexander Immer, Richard E. Turner, Frank Schneider, and Philipp Hennig. Kronecker-Factored Approximate Curvature for modern neural network architectures. In *NeurIPS*, 2023.

Vladimir Feinberg, Xinyi Chen, Y. Jennifer Sun, Rohan Anil, and Elad Hazan. Sketchy: Memory-efficient adaptive regularization with frequent directions. In *NeurIPS*, 2023.

Kai-Xin Gao, Xiaolei Liu, Zheng-Hai Huang, Min Wang, Zidong Wang, Dachuan Xu, and F. Yu. A trace-restricted Kronecker-factored approximation to natural gradient. In *AAAI Conference on Artificial Intelligence*, 2020.

Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. Fast approximate natural gradient descent in a Kronecker-factored eigenbasis. In *NeurIPS*, 2018.

Donald Goldfarb, Yi Ren, and Achraf Bahamou. Practical quasi-Newton methods for training deep neural networks. *Advances in Neural Information Processing Systems*, 33:2386–2396, 2020.

Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu,

Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The Llama 3 herd of models. arXiv 2407.21783, 2024.

Roger B. Grosse and James Martens. A Kronecker-factored approximate Fisher matrix for convolution layers. In *ICML*, 2016.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *ICML*, 2018.

Bobby He, Lorenzo Noci, Daniele Paliotta, Imanol Schlag, and Thomas Hofmann. Understanding and minimising outlier features in neural network training. In *NeurIPS*, 2024.

Tom Heskes. On "natural" learning and pruning in multilayered perceptrons. *Neural Computation*, 12(4), 2000.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. arXiv 2005.00687, 2021.

Satoki Ishikawa and Rio Yokota. When does second-order optimization speed up training? In *The Second Tiny Papers Track at ICLR*, 2024.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Priya Kasimbeg, Frank Schneider, Runa Eschenhagen, Juhan Bae, Chandramouli Shama Sastry, Mark Saroufim, Boyuan Feng, Less Wright, Edward Z. Yang, Zachary Nado, Sourabh Medapati, Philipp Hennig, Michael Rabbat, and George E. Dahl. Accelerating neural network training: An analysis of the AlgoPerf competition. In *ICLR*, 2025.

Nitish Shirish Keskar and Albert S Berahas. adaqn: An adaptive quasi-Newton algorithm for training RNNs. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I 16*, pp. 1–16. Springer, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Florian Knoll, Jure Zbontar, Anuroop Sriram, Matthew J. Muckley, Mary Bruno, Aaron Defazio, Marc Parente, Krzysztof J. Geras, Joe Katsnelson, Hersh Chandarana, Zizhao Zhang, Michal Drozdzalv, Adriana Romero, Michael Rabbat, Pascal Vincent, James Pinkerton, Duo Wang, Nafissa Yakubova, Erich Owens, C. Lawrence Zitnick, Michael P. Recht, Daniel K. Sodickson, and Yvonne W. Lui. fastMRI: A publicly available raw k-space and DICOM dataset of knee images for accelerated MR image reconstruction using machine learning. *Radiology: Artificial Intelligence*, 2(1):e190007, 2020. doi: 10.1148/ryai.2020190007. PMID: 32076662.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

Frederik Kunstner, Lukas Balles, and Philipp Hennig. Limitations of the empirical Fisher approximation for natural gradient descent. In *NeurIPS*, 2019.

Xi-Lin Li. Preconditioned stochastic gradient descent. *IEEE Transactions on Neural Networks and Learning Systems*, 29(5), 2018.

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. In *ICML*, 2024a.

Wu Lin, Felix Dangel, Runa Eschenhagen, Kirill Neklyudov, Agustinus Kristiadi, Richard E. Turner, and Alireza Makhzani. Structured inverse-free natural gradient: Memory-efficient & numerically-stable KFAC for large neural nets. In *ICML*, 2024b.

Wu Lin, Scott C. Lowe, Felix Dangel, Runa Eschenhagen, Zikun Xu, and Roger B. Grosse. Understanding and improving Shampoo and SOAP via Kullback-Leibler minimization. arXiv 2509.03378, 2025.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for llm training. arXiv 2502.16982, 2025.

Xialei Liu, Marc Masana, Luis Herranz, Joost van de Weijer, Antonio M. López, and Andrew D. Bagdanov. Rotate your networks: Better weight consolidation and less catastrophic forgetting. *ICPR*, 2018.

Lucas Maes, Tianyue H. Zhang, Alexia Jolicoeur-Martineau, Ioannis Mitliagkas, Damien Scieur, Simon Lacoste-Julien, and Charles Guille-Escuret. Understanding Adam requires better rotation dependent assumptions. arXiv 2410.19964, 2024.

James Martens. Deep learning via Hessian-free optimization. In *ICML*, 2010.

James Martens. New insights and perspectives on the natural gradient method. *JMLR*, 21(146), 2014.

James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *ICML*, 2015.

James Martens, Jimmy Ba, and Matt Johnson. Kronecker-factored curvature approximations for recurrent neural networks. In *ICLR*, 2018.

Depen Morwani, Itai Shapira, Nikhil Vyas, Eran Malach, Sham Kakade, and Lucas Janson. A new perspective on Shampoo's preconditioner. In *ICLR*, 2025.

Lucas Nestler. HeavyBall, 2025. URL https://github.com/HomebrewML/HeavyBall/.

Kazuki Osawa, Satoki Ishikawa, Rio Yokota, Shigang Li, and Torsten Hoefler. ASDL: A unified interface for gradient preconditioning in PyTorch. arXiv 2305.04684, 2023a.

Kazuki Osawa, Shigang Li, and Torsten Hoefler. PipeFisher: Efficient training of large language models using pipelining and Fisher information matrices. In Dawn Song, Michael Carbin, and Tianqi Chen (eds.), *MLSys*, 2023b.

J. Gregory Pauloski, Qi Huang, Lei Huang, Shivaram Venkataraman, Kyle Chard, Ian T. Foster, and Zhao Zhang. KAISA: an adaptive second-order optimizer framework for deep neural networks. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC21)*, 2021.

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained LMOs. arXiv 2502.07529, 2025.

Omead Pooladzandi and Xi-Lin Li. Curvature-informed SGD via general purpose lie-group preconditioners. arXiv 2402.04553, 2024.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv 1910.10683, 2023.

Yi Ren and Donald Goldfarb. Tensor normal training for deep learning models. In *NeurIPS*, 2021.

Thomas Robert, Mher Safaryan, Ionut-Vlad Modoranu, and Dan Alistarh. LDAdam: Adaptive optimization from low-dimensional gradient statistics. arXiv 2410.16103, 2024.

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. arXiv 2309.06497, 2023.

DiJia Su, Andrew Gu, Jane Xu, Yuandong Tian, and Jiawei Zhao. GaLore 2: Large-scale LLM pre-training by gradient low-rank projection. arXiv 2504.20437, 2025.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Mujin Kwun, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. SOAP: Improving and stabilizing Shampoo using Adam. In *ICLR*, 2025a.

Nikhil Vyas, Rosie Zhao, Depen Morwani, Mujin Kwun, and Sham Kakade. Improving SOAP using iterative whitening and Muon. https://nikhilvyas.github.io/SOAP_Muon.pdf, 2025b.

Evan Walters, Omead Pooladzandi, and Xi-Lin Li. kron_torch, 2025. URL https://github.com/evanatyourservice/kron_torch.

Sifan Wang, Ananyae Kumar Bhartari, Bowen Li, and Paris Perdikaris. Gradient alignment in physics-informed neural networks: A second-order optimization perspective. arXiv 2502.00604, 2025.

Sike Wang, Pan Zhou, Jia Li, and Hua Huang. 4-bit Shampoo for memory-efficient network training. In *NeurIPS*, 2024.

Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. ConvNeXt V2: Co-designing and scaling ConvNets with masked autoencoders. arXiv 2301.00808, 2023.

Shuo Xie, Tianhao Wang, Sashank Reddi, Sanjiv Kumar, and Zhiyuan Li. Structured preconditioners in adaptive optimization: A unified analysis. arXiv 2503.10537, 2025.

Jure Zbontar, Florian Knoll, Anuroop Sriram, Tullie Murrell, Zhengnan Huang, Matthew J. Muckley, Aaron Defazio, Ruben Stern, Patricia Johnson, Mary Bruno, Marc Parente, Krzysztof J. Geras, Joe Katsnelson, Hersh Chandarana, Zizhao Zhang, Michal Drozdzal, Adriana Romero, Michael Rabbat, Pascal Vincent, Nafissa Yakubova, James Pinkerton, Duo Wang, Erich Owens, C. Lawrence Zitnick, Michael P. Recht, Daniel K. Sodickson, and Yvonne W. Lui. fastMRI: An open dataset and benchmarks for accelerated MRI. arXiv 1811.08839, 2019.

Thomas T. Zhang, Behrad Moniri, Ansh Nagwekar, Faraz Rahman, Anton Xue, Hamed Hassani, and Nikolai Matni. On the concurrence of layer-wise preconditioning methods and provable feature learning. In *ICML*, 2025a.

Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Diederik P. Kingma, Yinyu Ye, Zhi-Quan Luo, and Ruoyu Sun. Adam-mini: Use fewer learning rates to gain more. In *ICLR*, 2025b.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. arXiv 2403.03507, 2024.

Table 2: Shampoo variants and their properties.

Shampoo variant	Justified from approximation perspective	Matches grafting in steps	Matches grafting in compute & memory	Learning rate transfer
grafting (Adam)	✗	✓	✓	✓
$\boldsymbol{C}^{\text{Shampoo}}$	✗	✗	✗	✗
$S^{-1}\boldsymbol{C}^{\text{Shampoo}^2}$	✓	✓	✗	?
$\boldsymbol{C}^{\text{EShampoo}}$	✓	✓	✓	✓

A Connecting full-matrix AdaGrad to the Fisher

We can also consider a more general class of preconditioned stochastic gradient methods with other choices of $\boldsymbol{C}_t$ and $p > 0$ that update the parameters at each iteration t by Equation (2). For example, one could choose $\boldsymbol{C}_t$ as approximations of the Hessian $\nabla^2\mathcal{L}(\boldsymbol{\theta})$ via subsampling or secant approximation with $p = 1$, which yields the class of *stochastic Newton* or *quasi-Newton* methods (Keskar & Berahas, 2016; Bollapragada et al., 2018; Berahas et al., 2020; Goldfarb et al., 2020). Another common choice is the Fisher information matrix $\boldsymbol{F}_t = \mathbb{E}_{\hat{\boldsymbol{y}}\sim f_{\boldsymbol{\theta}}(\boldsymbol{x}),\boldsymbol{x}\sim p_{\mathcal{D}}(\boldsymbol{x})}\left[\nabla_{\boldsymbol{\theta}}\ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}),\hat{\boldsymbol{y}})\nabla_{\boldsymbol{\theta}}\ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}),\hat{\boldsymbol{y}})^\intercal\right]$, which yields the *natural gradient* or, for common loss functions, *generalized Gauss-Newton method* with $p = 1$ (Amari, 1998; Martens, 2014). A summary of different methods is provided in Bottou et al. (2018).

By instead summing over per-sample gradient outer products, $\hat{\boldsymbol{A}}_t$ can be connected to the empirical Fisher. In general, the empirical Fisher is not expected to be a good approximation of the Fisher information matrix (Kunstner et al., 2019). However, by replacing the accumulation with an expectation over the conditional distribution given by the model $f_{\boldsymbol{\theta}}$, full-matrix AdaGrad can be made equivalent to the Fisher. This equivalence reveals a natural extension of Shampoo to approximate the Fisher, called Tensor Normal Training (Ren & Goldfarb, 2021, TNT), and is closely related to the K-FAC preconditioner (Anil et al., 2020). In fact, both approximations are exactly equivalent to the (block-diagonal) Fisher for simple cases such as deep linear networks (also with weight sharing) and mean squared error loss (Bernacchia et al., 2018; Eschenhagen et al., 2023; Morwani et al., 2025). The modifications in TNT deviate from the original Shampoo update (Gupta et al., 2018) because it was motivated by upper bounds to full-matrix AdaGrad in its non-smooth, convex regret analysis, rather than this Fisher approximation perspective.

B Algorithms pseudocode

In this section, we provide the pseudocode for all algorithms, including idealized (Algorithm 1) and practical (Algorithm 2) eigenvalue-corrected Shampoo, Shampoo with Adam grafting (Algorithm 3), and the adaptive warm-started QR algorithm (Algorithm 4). Algorithm 1 and Algorithm 3 present simplified versions of the algorithm ignoring common modifications like momentum or an exponential moving average over the stochastic gradient, bias corrections, and weight decay.

Note that different instances of eigenvalue-corrected Shampoo can be employed by changing the exponential moving average in Equation (12), eigenbasis computation in Equation (13), or eigenvalue correction update in Equation (14). SOAP delays the update of the eigenbasis until after the update step, approximates the eigenvalue correction (Equation (14)) by

$$\boldsymbol{D}_t = \beta_2\boldsymbol{D}_{t-1} + (1-\beta_2)\tilde{\boldsymbol{G}}_t^{\odot 2}, \tag{19}$$

and uses a single iteration of the warm-started simultaneous iteration. Additionally, there is a discrepancy between the SOAP algorithm presented in the paper and the official implementation: in the paper, the exponential moving average of the gradient $\boldsymbol{G}_t$ is used (line 4 of Algorithm 3 in Vyas et al. (2025a)), whereas the implementation computes the exponential moving average over the rotated gradient $\tilde{\boldsymbol{G}}_t$.[10] This can be interpreted as running Adam in Shampoo's eigenspace, which shares similarities to the GaLore algorithm (Zhao et al., 2024; Su et al., 2025).

[10] See https://github.com/nikhilvyas/SOAP/blob/f42d296cb4146a67fbe811371e6badb9a39cc54d/soap.py#L167.

Algorithm 1 Idealized eigenvalue-corrected Shampoo pseudocode

Require: Parameter $\boldsymbol{W}_1 \in \mathbb{R}^{m\times n}$, learning rate $\alpha_t > 0$, $\epsilon > 0$.

Initialize $\boldsymbol{L}_0 = \boldsymbol{0} \in \mathbb{R}^{m\times m}$, $\boldsymbol{R}_0 = \boldsymbol{0} \in \mathbb{R}^{n\times n}$, and $\boldsymbol{D} = \boldsymbol{0} \in \mathbb{R}^{m\times n}$.

for $t = 1, ..., T$ **do**

  Compute (mini-batch) stochastic gradient: $\boldsymbol{G}_t = \nabla_\theta \ell(f_{\boldsymbol{\theta}_t}(\boldsymbol{x}), \boldsymbol{y})$.

  Update factor matrices:

$$\boldsymbol{L}_t = \beta_2 \boldsymbol{L}_{t-1} + (1-\beta_2)\boldsymbol{G}_t\boldsymbol{G}_t^\intercal, \quad \boldsymbol{R}_t = \beta_2 \boldsymbol{R}_{t-1} + (1-\beta_2)\boldsymbol{G}_t^\intercal\boldsymbol{G}_t. \tag{12}$$

  Compute orthonormal eigenbasis of the factor matrices:

$$\boldsymbol{Q}_{\boldsymbol{L}_t} = \text{eigvec}(\boldsymbol{L}_t), \quad \boldsymbol{Q}_{\boldsymbol{R}_t} = \text{eigvec}(\boldsymbol{R}_t). \tag{13}$$

  Transform gradient basis: $\tilde{\boldsymbol{G}}_t = \boldsymbol{Q}_{\boldsymbol{L}_t}^\intercal \boldsymbol{G}_t \boldsymbol{Q}_{\boldsymbol{R}_t}$.

  Compute or update eigenvalue correction:

$$\boldsymbol{D}_t^* = \underset{\boldsymbol{D}\in\mathbb{R}^{m\times n}}{\arg\min} \|\boldsymbol{A}_t - (\boldsymbol{Q}_{\boldsymbol{R}_t} \otimes \boldsymbol{Q}_{\boldsymbol{L}_t})\,\text{diag}(\text{vec}(\boldsymbol{D}))(\boldsymbol{Q}_{\boldsymbol{R}_t} \otimes \boldsymbol{Q}_{\boldsymbol{L}_t})\|_F. \tag{14}$$

  Compute update: $\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \alpha_t \boldsymbol{Q}_{\boldsymbol{L}_t}(\tilde{\boldsymbol{G}}_t \oslash (\sqrt{\boldsymbol{D}_t^*} + \epsilon \boldsymbol{1}\boldsymbol{1}^\intercal))\boldsymbol{Q}_{\boldsymbol{R}_t}$.

end for

Algorithm 2 EShampoo pseudocode (as implemented for the experiments)

Require: Parameter $\boldsymbol{W}_1 \in \mathbb{R}^{m\times n}$, learning rate $\alpha_t > 0$, $\epsilon > 0, \beta_1, \beta_2 \in [0, 1)$, weight decay $\lambda \geq 0$, eigenbasis computation frequency $F \in \mathbb{N}$, and threshold $\tau \in [0, 1]$ for Equation (11) and Algorithm 4.

Initialize $\boldsymbol{M}_0 = \boldsymbol{0} \in \mathbb{R}^{m\times n}, \boldsymbol{L}_0 = \boldsymbol{0} \in \mathbb{R}^{m\times m}$, $\boldsymbol{R}_0 = \boldsymbol{0} \in \mathbb{R}^{n\times n}, \boldsymbol{Q}_{\boldsymbol{L}_0} = \mathbf{I}_m, \boldsymbol{Q}_{\boldsymbol{R}_0} = \mathbf{I}_n$, and $\boldsymbol{D}_0 = \boldsymbol{0} \in \mathbb{R}^{m\times n}$.

for $t = 1, ..., T$ **do**

  Compute (mini-batch) stochastic gradient: $\boldsymbol{G}_t = \nabla_\theta \ell(f_{\boldsymbol{\theta}_t}(\boldsymbol{x}), \boldsymbol{y})$.

  Compute exponential moving average of the gradient: $\boldsymbol{M}_t = \beta_1 \boldsymbol{M}_{t-1} + (1-\beta_1)\,\boldsymbol{G}_t$.

  Update factor matrices:

$$\boldsymbol{L}_t = \beta_2 \boldsymbol{L}_{t-1} + (1-\beta_2)\boldsymbol{G}_t\boldsymbol{G}_t^\intercal, \quad \boldsymbol{R}_t = \beta_2 \boldsymbol{R}_{t-1} + (1-\beta_2)\boldsymbol{G}_t^\intercal\boldsymbol{G}_t. \tag{15}$$

  if $t \mod F = 0$ and Equation (11) **then**

    Compute eigenbasis of factor matrices (e.g. with `torch.linalg.eigh` or Algorithm 4):

$$\boldsymbol{Q}_{\boldsymbol{L}_t} = \text{eigvec}(\boldsymbol{L}_t/(1-\beta_2^t)), \quad \boldsymbol{Q}_{\boldsymbol{R}_t} = \text{eigvec}(\boldsymbol{R}_t/(1-\beta_2^t)). \tag{16}$$

  else

$$\boldsymbol{Q}_{\boldsymbol{L}_t} = \boldsymbol{Q}_{\boldsymbol{L}_{t-1}}, \quad \boldsymbol{Q}_{\boldsymbol{R}_t} = \boldsymbol{Q}_{\boldsymbol{R}_{t-1}}. \tag{17}$$

  end if

  Transform gradient basis: $\tilde{\boldsymbol{G}}_t = \boldsymbol{Q}_{\boldsymbol{L}_t}^\intercal \boldsymbol{G}_t \boldsymbol{Q}_{\boldsymbol{R}_t}$.

  Compute or update eigenvalue correction: $\boldsymbol{D}_t = \beta_2 \boldsymbol{D}_{t-1} + (1-\beta_2)\tilde{\boldsymbol{G}}_t^{\odot 2}$.

  Perform bias correction:

$$\tilde{\boldsymbol{M}}_t = \boldsymbol{M}_t/(1-\beta_1^t), \quad \tilde{\boldsymbol{D}}_t = \boldsymbol{D}_t/(1-\beta_2^t).$$

  Compute parameter update:

$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \alpha_t \left(\boldsymbol{Q}_{\boldsymbol{L}_t} \left(\boldsymbol{Q}_{\boldsymbol{L}_t}^\intercal \tilde{\boldsymbol{M}}_t \boldsymbol{Q}_{\boldsymbol{R}_t} \oslash \left(\sqrt{\tilde{\boldsymbol{D}}_t} + \epsilon \boldsymbol{1}\boldsymbol{1}^\intercal \right) \right) \boldsymbol{Q}_{\boldsymbol{R}_t}^\intercal + \lambda \boldsymbol{W}_t \right). \tag{18}$$

end for

Algorithm 3 Shampoo with Adam grafting pseudocode

Require: Parameter $\boldsymbol{W}_1 \in \mathbb{R}^{m\times n}$, learning rate $\alpha_t > 0$, exponential moving average constant $\beta_2 \in (0,1)$, $\epsilon > 0$.

Initialize $\boldsymbol{L}_0 = \boldsymbol{0} \in \mathbb{R}^{m\times m}$, $\boldsymbol{R}_0 = \boldsymbol{0} \in \mathbb{R}^{n\times n}$, and $\boldsymbol{D} = \boldsymbol{0} \in \mathbb{R}^{m\times n}$.

for $t = 1, ..., T$ **do**

  Compute (mini-batch) stochastic gradient: $\boldsymbol{G}_t = \nabla_\theta \ell(f_{\boldsymbol{\theta}_t}(\boldsymbol{x}), \boldsymbol{y})$.

  Update factor matrices:

$$\boldsymbol{L}_t = \beta_2 \boldsymbol{L}_{t-1} + (1-\beta_2)\boldsymbol{G}_t\boldsymbol{G}_t^\intercal, \quad \boldsymbol{R}_t = \beta_2 \boldsymbol{R}_{t-1} + (1-\beta_2)\boldsymbol{G}_t^\intercal\boldsymbol{G}_t.$$

  Update Adam grafting state: $\boldsymbol{D}_t = \beta_2 \boldsymbol{D}_{t-1} + (1-\beta_2)\boldsymbol{G}_t^{\odot 2}$.

  Compute matrix root inverse of the factor matrices:

$$\boldsymbol{L}_t^{-1/4} = \text{rootinv}(\boldsymbol{L}_t), \quad \boldsymbol{R}_t^{-1/4} = \text{rootinv}(\boldsymbol{R}_t).$$

  Compute update: $\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \alpha_t \frac{\|-\boldsymbol{G}_t \oslash (\sqrt{\boldsymbol{D}_t} + \epsilon \boldsymbol{1}\boldsymbol{1}^T)\|_F}{\|-\boldsymbol{L}_t^{-1/4}\boldsymbol{G}\boldsymbol{R}_t^{-1/4}\|_F} \boldsymbol{L}_t^{-1/4}\boldsymbol{G}_t\boldsymbol{R}_t^{-1/4}$.

end for

Algorithm 4 Warm-started QR iteration with relative error termination criterion.

Require: Matrix $\boldsymbol{L} = \boldsymbol{L}_t$, previous eigenbasis $\hat{\boldsymbol{Q}} = \hat{\boldsymbol{Q}}_{\boldsymbol{L}_{t-1}}$, relative tolerance $\tau \in [0,1)$, maximum number of iterations I.

$i \leftarrow 0$

$\hat{\boldsymbol{\Lambda}} \leftarrow \hat{\boldsymbol{Q}}^\intercal \boldsymbol{L} \hat{\boldsymbol{Q}}$

while $\|\hat{\boldsymbol{\Lambda}} - \text{diag}(\hat{\boldsymbol{\Lambda}})\|_F \leq \tau \|\hat{\boldsymbol{\Lambda}}\|_F$ and $i < I$ **do**

  $\boldsymbol{Q}, \boldsymbol{R} \leftarrow \texttt{QR}(\hat{\boldsymbol{\Lambda}})$

  $\hat{\boldsymbol{\Lambda}} \leftarrow \boldsymbol{R}\boldsymbol{Q}$

  $\hat{\boldsymbol{Q}} \leftarrow \hat{\boldsymbol{Q}}\boldsymbol{Q}$

  $i \leftarrow i + 1$

end while

return $\hat{\boldsymbol{Q}}$, $\hat{\boldsymbol{\Lambda}}$

In contrast, our implementation of EShampoo does not delay the update of the preconditioner, uses the same approximation of the eigenvalue correction as SOAP, uses `torch.linalg.eigh` for the eigendecomposition (unless indicated otherwise), and computes the exponential moving average over the gradient $\boldsymbol{G}_t$. Both algorithms use bias corrections and (decoupled) weight decay. We provide the pseudocode for EShampoo as it was implemented for our experiments in Algorithm 2. If $F \neq 1$, the algorithm reduces to AdamW until iteration $t = F$.

While we do not present experimental results here, Algorithm 4 or Equation (11) for `eigh` can also be used in Shampoo without an eigenvalue correction. This implementation stores the previous eigendecomposition of the Kronecker factors instead of the previous root-inverse Kronecker factors. When we skip an eigendecomposition, we maintain the previous eigenbasis and replace the previous eigenvalues with the estimated eigenvalues $\text{diag}(\hat{\boldsymbol{\Lambda}}_{\boldsymbol{L}_t}) = \text{diag}(\boldsymbol{Q}_{\boldsymbol{L}_{t-1}}^\intercal \boldsymbol{L}_t \boldsymbol{Q}_{\boldsymbol{L}_{t-1}})$ used in Equation (11). Alternatively, one can compute an exponential moving average over these estimated eigenvalues as done in KL-Shampoo (Lin et al., 2025), which also removes the need for grafting like the eigenvalue correction in SOAP and EShampoo (c.f. Section 3).

C Proofs

Lemma 1. *Let* $\boldsymbol{U} = \boldsymbol{Q}_{\boldsymbol{L}}(\boldsymbol{D}^{\odot -p} \odot (\boldsymbol{Q}_{\boldsymbol{L}}^\intercal \boldsymbol{G} \boldsymbol{Q}_{\boldsymbol{R}}))\boldsymbol{Q}_{\boldsymbol{R}}^\intercal \in \mathbb{R}^{m\times n}$ *be the generalized eigendecomposed Kronecker-factored update given by orthogonal matrices* $\boldsymbol{Q}_{\boldsymbol{L}} \in \mathbb{R}^{m\times m}$, $\boldsymbol{Q}_{\boldsymbol{R}} \in \mathbb{R}^{n\times n}$, *and dense scaling matrix* $\boldsymbol{D} \in \mathbb{R}^{m\times n}$. *Then we have:*

$$(\max_{i,j} \boldsymbol{D}_{i,j})^{-p} ||\boldsymbol{G}||_F \leq ||\boldsymbol{U}||_F \leq (\min_{i,j} \boldsymbol{D}_{i,j})^{-p} ||\boldsymbol{G}||_F. \tag{20}$$

Proof. Since the Frobenius norm of a matrix is invariant to orthogonal transformations and the entries of $\boldsymbol{D}$ are bounded, i.e., $\min_{i,j} \boldsymbol{D}_{i,j} \leq \boldsymbol{D}_{i,j} \leq \max_{i,j} \boldsymbol{D}_{i,j}$, we can show that

$$\begin{aligned}
\|\boldsymbol{U}\|_F &= \|\boldsymbol{Q_L}(\boldsymbol{D}^{\odot -p} \odot (\boldsymbol{Q_L^\intercal} \boldsymbol{G} \boldsymbol{Q_R}))\boldsymbol{Q_R^\intercal}\|_F \\
&= \|\boldsymbol{D}^{\odot -p} \odot (\boldsymbol{Q_L^\intercal} \boldsymbol{G} \boldsymbol{Q_R}))\|_F \\
&\in [(\max_{i,j} \boldsymbol{D}_{i,j})^{-p}, (\min_{i,j} \boldsymbol{D}_{i,j})^{-p}] \cdot \|\boldsymbol{Q_L^\intercal} \boldsymbol{G} \boldsymbol{Q_R}\|_F \\
&\in [(\max_{i,j} \boldsymbol{D}_{i,j})^{-p}, (\min_{i,j} \boldsymbol{D}_{i,j})^{-p}] \cdot \|\boldsymbol{G}\|_F.
\end{aligned}$$

□

Proposition 1. *Assume that $\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal]$ is symmetric positive definite. The magnitude of the updates for full-matrix Adam, diagonal Adam, and eigenvalue-corrected Shampoo are all bounded by the power of the extreme eigenvalues of full-matrix Adam:*

$$\lambda_{\max}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal])^{-p} \|\boldsymbol{G}\|_F \leq \|\boldsymbol{U}\|_F \leq \lambda_{\min}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal])^{-p} \|\boldsymbol{G}\|_F, \tag{21}$$

for all $p > 0$. However, under the simplifying assumption that $\mathbb{E}[\boldsymbol{G}] = \boldsymbol{0}$ and $\boldsymbol{G}_{i,j}$ is independent from $\boldsymbol{G}_{k,l}$ for $(i,j) \neq (k,l)$ and has bounded second moment, $\lambda_{\min}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal]) \leq \mathbb{E}[\boldsymbol{G}_{i,j}^2] \leq \lambda_{\max}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal])$ and Shampoo has dimension-dependent bounds:

$$m^{-p/2} n^{-p/2} \lambda_{\max}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal])^{-p} \|\boldsymbol{G}\|_F \leq \|\boldsymbol{U}\|_F \leq m^{-p/2} n^{-p/2} \lambda_{\min}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal])^{-p} \|\boldsymbol{G}\|_F. \tag{22}$$

Proof. Note that Equation (21) holds for full-matrix Adam since $\|\boldsymbol{U}\|_F = \|\boldsymbol{u}\|_2 = \|\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal]^{-p}\boldsymbol{g}\|_2 \in [\lambda_{\max}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal])^{-p}, \lambda_{\min}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal])^{-p}] \cdot \|\boldsymbol{g}\|_2$, where $\boldsymbol{u} = \operatorname{vec}(\boldsymbol{U})$ and $\boldsymbol{g} = \operatorname{vec}(\boldsymbol{G})$.

For diagonal Adam and eigenvalue-corrected Shampoo, it is sufficient to show that $\lambda_{\min}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal]) \leq \boldsymbol{D}_{i,j} \leq \lambda_{\max}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal])$ for all i, j and apply Lemma 1. To see this, note that $\boldsymbol{D}_{i,j}$ can be represented by a Rayleigh quotient, i.e.,

$$\begin{aligned}
\boldsymbol{D}_{i,j} &= \mathbb{E}[\boldsymbol{G}_{i,j}^2] = \boldsymbol{e}_{i,j}^\intercal \mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal] \boldsymbol{e}_{i,j} && \text{(Adam)} \\
\boldsymbol{D}_{i,j} &= \mathbb{E}[(\boldsymbol{Q}_L^\intercal \boldsymbol{G} \boldsymbol{Q}_R)_{i,j}^2] = \boldsymbol{e}_{i,j}^\intercal (\boldsymbol{Q}_R \otimes \boldsymbol{Q}_L)^\intercal \mathbb{E}[\boldsymbol{g}\boldsymbol{g}^T](\boldsymbol{Q}_R \otimes \boldsymbol{Q}_L)\boldsymbol{e}_{i,j} && \text{(EShampoo)}
\end{aligned}$$

where $\boldsymbol{e}_{i,j} = \operatorname{vec} \boldsymbol{E}_{i,j} \in \mathbb{R}^{mn}$ and $\boldsymbol{E}_{i,j} \in \mathbb{R}^{m \times n}$ with

$$(\boldsymbol{E}_{i,j})_{k,l} = \begin{cases} 1 & \text{if } (k,l) = (i,j) \\ 0 & \text{otherwise.} \end{cases}$$

Since $\|\boldsymbol{e}_{i,j}\|_2 = \|(\boldsymbol{Q}_R \otimes \boldsymbol{Q}_L)\boldsymbol{e}_{i,j}\|_2 = 1$, the desired bound by the extreme eigenvalues follows from the Courant–Fischer–Weyl min-max theorem.

To prove Equation (22), observe that under independence between components of the gradient $\boldsymbol{G}_{i,j}$, the preconditioner for full-matrix Adam $\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal]$ is a diagonal matrix whose diagonal entries consist of $\mathbb{E}[\boldsymbol{G}_{i,j}^2]$ for all i, j. Hence, $\mathbb{E}[\boldsymbol{G}_{i,j}^2]$ is bounded by the minimum and maximum eigenvalues, i.e., $\mathbb{E}[\boldsymbol{G}_{i,j}^2] \in \left[\lambda_{\min}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal]), \lambda_{\max}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal])\right]$.

Due to independence, $\mathbb{E}[\boldsymbol{G}\boldsymbol{G}^\intercal]$ and $\mathbb{E}[\boldsymbol{G}^\intercal\boldsymbol{G}]$ are also diagonal. Expanding their diagonal entries gives

$$\begin{aligned}
(\mathbb{E}[\boldsymbol{G}\boldsymbol{G}^\intercal])_{i,i} &= \sum_{j=1}^{n} \mathbb{E}[\boldsymbol{G}_{i,j}^2] \in n \cdot \left[\lambda_{\min}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal]), \lambda_{\max}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal])\right] && \forall\, i = 1, ..., m, \\
(\mathbb{E}[\boldsymbol{G}^\intercal\boldsymbol{G}])_{j,j} &= \sum_{i=1}^{m} \mathbb{E}[\boldsymbol{G}_{i,j}^2] \in m \cdot \left[\lambda_{\min}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal]), \lambda_{\max}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal])\right] && \forall\, j = 1, ..., n.
\end{aligned}$$

Therefore, the Shampoo preconditioner $(\mathbb{E}[\boldsymbol{G}\boldsymbol{G}^\intercal] \otimes \mathbb{E}[\boldsymbol{G}^\intercal\boldsymbol{G}])^{1/2}$ is also diagonal with eigenvalues lying in the interval $\left[m^{1/2}n^{1/2}\lambda_{\min}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal]), m^{1/2}n^{1/2}\lambda_{\max}(\mathbb{E}[\boldsymbol{g}\boldsymbol{g}^\intercal])\right]$. The desired result follows. □

Note that more general bounds can be derived by relaxing the assumption $\mathbb{E}[\boldsymbol{G}] = \boldsymbol{0}$, although the bounds are more complex and are not more conceptually informative.

D On the gap between the optimal and practical eigenvalue correction

Eigenvalue-corrected Shampoo shares similarities with memory-efficient optimizers such as GaLore, which apply Adam in a low-dimensional subspace spanned by the largest singular vectors of the gradient matrix (Zhao et al., 2024; Su et al., 2025). However, GaLore relies on the assumption that the gradient resides in a low-rank subspace that evolves gradually, allowing the same optimizer state to be used even as the subspace is updated. A recent method called LDAdam proposed by Robert et al. (2024) describes a projection-aware method that corrects the scaling matrix through both a projection-aware update and generalized error feedback mechanism when transitioning between subspaces to address this issue. We will see that the naive eigenvalue correction as used in SOAP suffers from similar limitations.

D.1 Optimal eigenvalue correction in Frobenius norm

Note that we can determine the optimal eigenvalue correction by minimizing its Frobenius norm approximation to full-matrix AdaGrad or Adam:

Proposition 2. *Given symmetric matrix $\boldsymbol{C} \in \mathbb{R}^{d\times d}$ and orthogonal matrix $\boldsymbol{Q} \in \mathbb{R}^{d\times d}$, the optimal eigenvalue correction $\boldsymbol{D}^*$ that minimizes the Frobenius norm distance is given by:*

$$\boldsymbol{D}^* := \operatorname{diag}(\boldsymbol{Q}^\intercal \boldsymbol{C} \boldsymbol{Q}) \in \arg\min_{\boldsymbol{D}\in\mathbb{D}^d} ||\boldsymbol{C} - \boldsymbol{Q}\boldsymbol{D}\boldsymbol{Q}^\intercal||_{\mathrm{F}}.^{11} \tag{23}$$

The exact expression for $\boldsymbol{D}^*$ depends on the form of accumulation used for computing $\boldsymbol{C}$; note that the damping term is omitted here since it does not change the optimal solution. T denotes the current iteration.

1. *(Idealized) If $\boldsymbol{C} = \mathbb{E}\left[\boldsymbol{g}\boldsymbol{g}^\intercal\right]$, then $\boldsymbol{D}^* = \mathbb{E}\left[\operatorname{diag}(\boldsymbol{Q}^\intercal \boldsymbol{g})^{\odot 2}\right]$ (George et al., 2018).*
2. *(AdaGrad) If $\boldsymbol{C} = \sum_{t=1}^{T} \boldsymbol{g}_t \boldsymbol{g}_t^\intercal$, then $\boldsymbol{D}_T^* = \sum_{t=1}^{T} \operatorname{diag}(\boldsymbol{Q}_T^\intercal \boldsymbol{g}_t)^{\odot 2}$.*
3. *(Adam) If $\boldsymbol{C} = (1-\beta_2)\sum_{t=1}^{T} \beta_2^{T-t} \boldsymbol{g}_t \boldsymbol{g}_t^\intercal$, then $\boldsymbol{D}_T^* = (1-\beta_2)\sum_{t=1}^{T} \beta_2^{T-t} \operatorname{diag}(\boldsymbol{Q}_T^\intercal \boldsymbol{g}_t)^{\odot 2}$. Note that $\boldsymbol{C}$ can be generated recursively via an exponential moving average $\boldsymbol{C}_T = \beta_2 \boldsymbol{C}_{T-1} + (1-\beta_2)\boldsymbol{g}_T \boldsymbol{g}_T^\intercal$ with $\boldsymbol{C}_0 = \boldsymbol{0} \in \mathbb{R}^{mn\times mn}$.*

Proof. (Informal.) Since $\boldsymbol{Q}$ is orthogonal, $\|\boldsymbol{C} - \boldsymbol{Q}\boldsymbol{D}\boldsymbol{Q}^\intercal\|_F = \|\boldsymbol{Q}^\intercal \boldsymbol{C}\boldsymbol{Q} - \boldsymbol{D}\|_F$, with $\boldsymbol{D}$ diagonal. Therefore, the optimal solution has the form $\boldsymbol{D}^* = \operatorname{diag}(\boldsymbol{Q}^\intercal \boldsymbol{C}\boldsymbol{Q})$. Each case then follows by observing that $\boldsymbol{C}$ is the expectation or weighted sum of $\boldsymbol{g}\boldsymbol{g}^\intercal$, and passing $\boldsymbol{Q}$ into the sum or expectation. □

While $\boldsymbol{D}^*$ is optimal in this sense, it does not guarantee anything regarding the similarity of the (root) inverse of the approximation. Using Proposition 2, we can establish the following corollary.

Corollary 1. *The optimal eigenvalue correction yields a tighter Frobenius norm approximation than Shampoo and CASPR, i.e.,*

$$||\boldsymbol{C} - \boldsymbol{Q}\boldsymbol{D}^*\boldsymbol{Q}^\intercal||_{\mathrm{F}} \leq \min\{||\boldsymbol{C} - \boldsymbol{C}^{\mathrm{Shampoo}}||_{\mathrm{F}}, ||\boldsymbol{C} - \boldsymbol{C}^{\mathrm{CASPR}}||_{\mathrm{F}}\},$$

where $\boldsymbol{C}^{\mathrm{CASPR}}$ is the preconditioner used by the CASPR algorithm (Duvvuri et al., 2024).

Proof. (Informal.) The first inequality trivially follows from Proposition 2 since by definition the eigenvectors of Shampoo are equal to $\boldsymbol{Q} = (\boldsymbol{Q}_{\boldsymbol{R}} \otimes \boldsymbol{Q}_{\boldsymbol{L}})^\intercal$. The second inequality follows from Proposition 2 together with Lemma 3.4 in Duvvuri et al. (2024), which states that the eigenvectors of $\boldsymbol{C}^{\mathrm{Shampoo}}$ and $\boldsymbol{C}^{\mathrm{CASPR}}$ are identical. □

This means that using the optimal eigenvalue correction (with respect to Frobenius norm approximation) yields a tighter approximation to the full-matrix quantity compared to Shampoo or CASPR. A remaining open theoretical question is whether a tighter Frobenius norm approximation can yield a tighter regret bound compared to Shampoo and CASPR. For example, one can obtain a tighter regret bound than Shampoo by only considering one-sided Shampoo, which is not generally a better approximation to full-matrix AdaGrad (Xie et al., 2025; An et al., 2025).

[11] $\mathbb{D}^d$ denotes the set of $d \times d$ diagonal matrices.

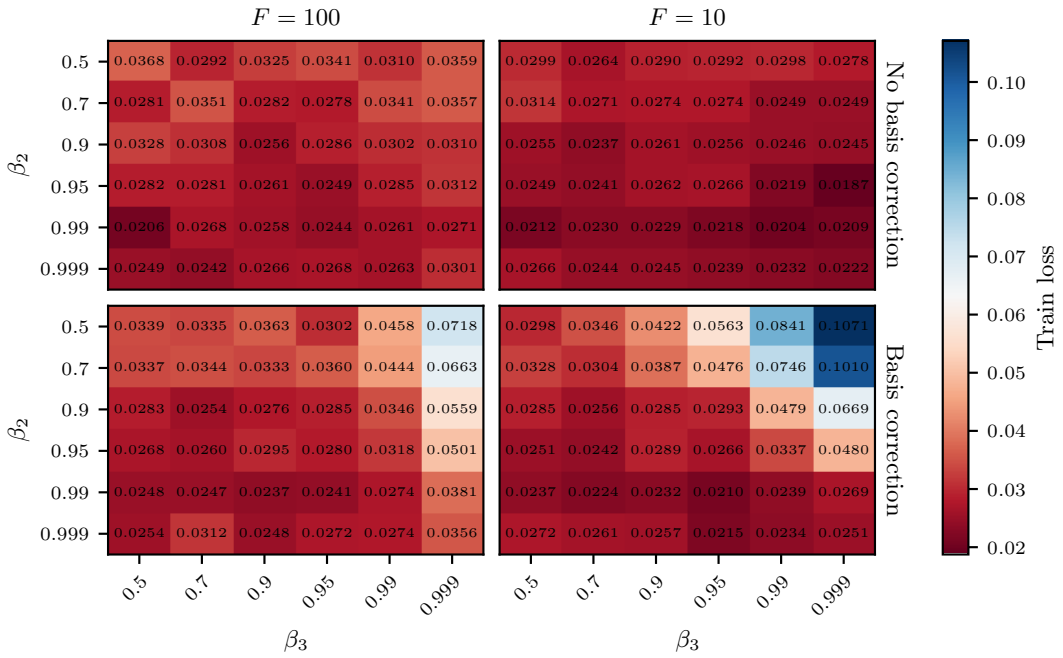

Figure 6: Decoupling the exponential moving average for the eigenbasis (β_2) and eigenvalues (β_3) for different eigendecomposition frequencies (F). We can observe a remarkable invariance to the choice of β_2 and β_3. Interestingly, the correction for the change of bases seems to *hurt* performance overall and especially for low β_2, high β_3, and low F values – a pattern that we might expect *without* the correction.

D.2 Basis-aware eigenvalue correction

Despite its optimality, computing $\boldsymbol{D}^*$ for cases 2 and 3 in Proposition 2 is not feasible in practice since we would have to store and transform prior gradients from *all previous iterations* with $\boldsymbol{Q}_T$, or have access to the full-matrix quantity $\boldsymbol{C}_T$. For simplicity, we will focus on case 3, although the results generalize without loss of generality.[12] In contrast to the optimal eigenvalue correction, which uses a fixed basis matrix $\boldsymbol{Q}_T$ based on the most recent statistics $\boldsymbol{L}_T$ and $\boldsymbol{R}_T$, SOAP uses potentially different basis matrices $\boldsymbol{Q}_t$ based on previous statistics $\boldsymbol{L}_t$ and $\boldsymbol{R}_t$ available at every step $t = 1, ..., T$:

$$\boldsymbol{D}^* = (1-\beta_2)\sum_{t=1}^{T}\beta_2^{T-t}\operatorname{diag}\left(\boldsymbol{Q}_T^\intercal \boldsymbol{g}_t\right)^{\odot 2} \approx (1-\beta_2)\sum_{t=1}^{T}\beta_2^{T-t}\operatorname{diag}\left(\boldsymbol{Q}_t^\intercal \boldsymbol{g}_t\right)^{\odot 2} =: \hat{\boldsymbol{D}}_T. \tag{24}$$

Intuitively, this means that the eigenvalue correction is accumulated inconsistently *across different coordinate systems*. This can lead to a mismatch when preconditioning $\boldsymbol{G}_T$, which is only transformed to the *current* coordinate system determined by $\boldsymbol{Q}_T$. When the basis remains approximately constant, i.e., $\boldsymbol{Q}_T \approx \boldsymbol{Q}_t$ for all t, Equation (24) can be a tight approximation and $\boldsymbol{D}^* \approx \boldsymbol{D}_T$.

The naive eigenvalue correction may be approximately correct when the basis is updated infrequently, but this can be a poor approximation if $\boldsymbol{Q}$ does not approximate the changing eigenbasis of $\boldsymbol{C}$, thereby increasing the approximation error. For case 3, the approximation becomes potentially milder because the terms in the sum in Equation (24) are down-weighted by β_2^{T-t} through the exponential moving average. Therefore, depending on β_2, the contribution from the eigenvalue correction statistic in previous coordinate systems might be negligible. However, this is not the case for case 2, where all terms are weighted equally.

[12] To address case 2, simply drop $1-\beta_2$ and β_2^{T-t}.

In order to address the theory-practice gap between the naive eigenvalue correction (used in SOAP and EShampoo) and the optimal correction in Frobenius norm, we first observe that

$$\begin{aligned}
\boldsymbol{D}^* &= (1-\beta_2)\sum_{t=1}^{T}\beta_2^{T-t}\operatorname{diag}\left(\boldsymbol{Q}^\intercal\boldsymbol{g}_t\right)^{\odot 2} \\
&= (1-\beta_2)\sum_{t=1}^{T}\beta_2^{T-t}\operatorname{diag}\left(\boldsymbol{Q}_T^\intercal\boldsymbol{g}_t\boldsymbol{g}_t^\intercal\boldsymbol{Q}_T\right) \\
&= \operatorname{diag}\left(\boldsymbol{Q}_T^\intercal\big((1-\beta_2)\sum_{t=1}^{T}\beta_2^{T-t}\boldsymbol{g}_t\boldsymbol{g}_t^\intercal\big)\boldsymbol{Q}_T\right) \\
&= \operatorname{diag}\left(\boldsymbol{Q}_T^\intercal\boldsymbol{C}_T\boldsymbol{Q}_T\right) \\
&=: \operatorname{diag}\left(\hat{\boldsymbol{C}}_T\right).
\end{aligned} \tag{25}$$

Since $\boldsymbol{Q}_T$ is orthogonal, we can recursively write

$$\begin{aligned}
\boldsymbol{D}^* = \operatorname{diag}\left(\hat{\boldsymbol{C}}_T\right) &= \operatorname{diag}\left(\boldsymbol{Q}_T^\intercal(\beta_2\boldsymbol{C}_{T-1}+(1-\beta_2)\boldsymbol{g}_T\boldsymbol{g}_T^\intercal)\boldsymbol{Q}_T\right) \\
&= \operatorname{diag}\left(\beta_2\boldsymbol{Q}_T^\intercal\boldsymbol{C}_{T-1}\boldsymbol{Q}_T+(1-\beta_2)\boldsymbol{Q}_T^\intercal\boldsymbol{g}_t\boldsymbol{g}_t^\intercal\boldsymbol{Q}_T\right) \\
&= \operatorname{diag}\left(\beta_2\boldsymbol{Q}_T^\intercal\boldsymbol{Q}_{T-1}\hat{\boldsymbol{C}}_{T-1}\boldsymbol{Q}_{T-1}^\intercal\boldsymbol{Q}_T+(1-\beta_2)\boldsymbol{Q}_T^\intercal\boldsymbol{g}_T\boldsymbol{g}_T^\intercal\boldsymbol{Q}_T\right) \\
&= \beta_2\operatorname{diag}\left(\boldsymbol{R}_{T,T-1}\hat{\boldsymbol{C}}_{T-1}\boldsymbol{R}_{T,T-1}^\intercal\right)+(1-\beta_2)\operatorname{diag}\left(\boldsymbol{Q}_T^\intercal\boldsymbol{g}_T\right)^{\odot 2},
\end{aligned} \tag{26}$$

where $\boldsymbol{R}_{T,T-1} := \boldsymbol{Q}_T^\intercal\boldsymbol{Q}_{T-1}$ is the transition matrix between different bases.

While this is the exact expression for our desired quantity, it requires keeping track of the full matrix $\hat{\boldsymbol{C}}_{T-1}$ to compute $\operatorname{diag}(\boldsymbol{R}_{T,T-1}\hat{\boldsymbol{C}}_{T-1}\boldsymbol{R}_{T,T-1}^\intercal)$, which is not tractable. Since we explicitly construct $\boldsymbol{Q}_{T-1}$ to be close to the best Kronecker-factored basis for $\boldsymbol{C}_{T-1}$ through the choice of the Shampoo preconditioner with $\boldsymbol{L}_{T-1}$ and $\boldsymbol{R}_{T-1}$, we make the additional assumption that $\hat{\boldsymbol{C}}_{T-1}$ is approximately diagonal, i.e., $\hat{\boldsymbol{C}}_{T-1} \approx \operatorname{diag}(\hat{\boldsymbol{C}}_{T-1}) = \operatorname{diag}(\boldsymbol{v}_{T-1}^{\text{corrected}})$.

Substituting this back into the recursive equation above, we have that

$$\begin{aligned}
\boldsymbol{D}^* = \operatorname{diag}\left(\hat{\boldsymbol{C}}_T\right) &\approx \beta_2\operatorname{diag}\left(\boldsymbol{R}_{T,T-1}\operatorname{diag}\left(\boldsymbol{v}_{T-1}^{\text{corrected}}\right)\boldsymbol{R}_{T,T-1}^\intercal\right)+(1-\beta_2)\operatorname{diag}\left(\boldsymbol{Q}_T^\intercal\boldsymbol{g}_T\right)^{\odot 2} \\
&= \beta_2\operatorname{diag}\left(\boldsymbol{R}_{T,T-1}^{\odot 2}\boldsymbol{v}_{T-1}^{\text{corrected}}\right)+(1-\beta_2)\operatorname{diag}\left(\boldsymbol{Q}_T^\intercal\boldsymbol{g}_T\right)^{\odot 2} \\
&= \operatorname{diag}\left(\boldsymbol{v}_T^{\text{corrected}}\right).
\end{aligned} \tag{27}$$

This gives a recursive update rule that, in contrast to the naive solution, accounts for the changes of bases between iterations. This also recovers the exact solution when $\boldsymbol{Q}_T = \boldsymbol{Q}_t$ for all t since $\boldsymbol{R}_{t,t-1} = \mathbf{I}$. This correction for the changing basis has a similar motivation to and resembles parts of the LDAdam algorithm (Robert et al., 2024).

We design an experiment to empirically test whether the implicit approximation in Equation (24) helps in practice, and find that interestingly, the correction appears to hurt performance in the settings where we would expect it to help, see Figure 6. A satisfying explanation of this phenomenon remains an open question.

E Additional experimental details and results

For all experiments we used $1\times$ NVIDIA A100 80GB GPU per run, with the exception of the ImageNet ViT experiments, for which we used $4\times$ NVIDIA A100 80GB GPUs per run. All of the experiments were conducted on an internal compute cluster and we estimate that the total required compute was around 1440 GPU hours or 60 days. We ran additional exploratory experiments not reported in this paper which increases the total compute cost of the project. The implementation of EShampoo and all other Shampoo variants considered here including Algorithm 4 and Equation (11) for `eigh` is available at `https://github.com/facebookresearch/optimizers`.

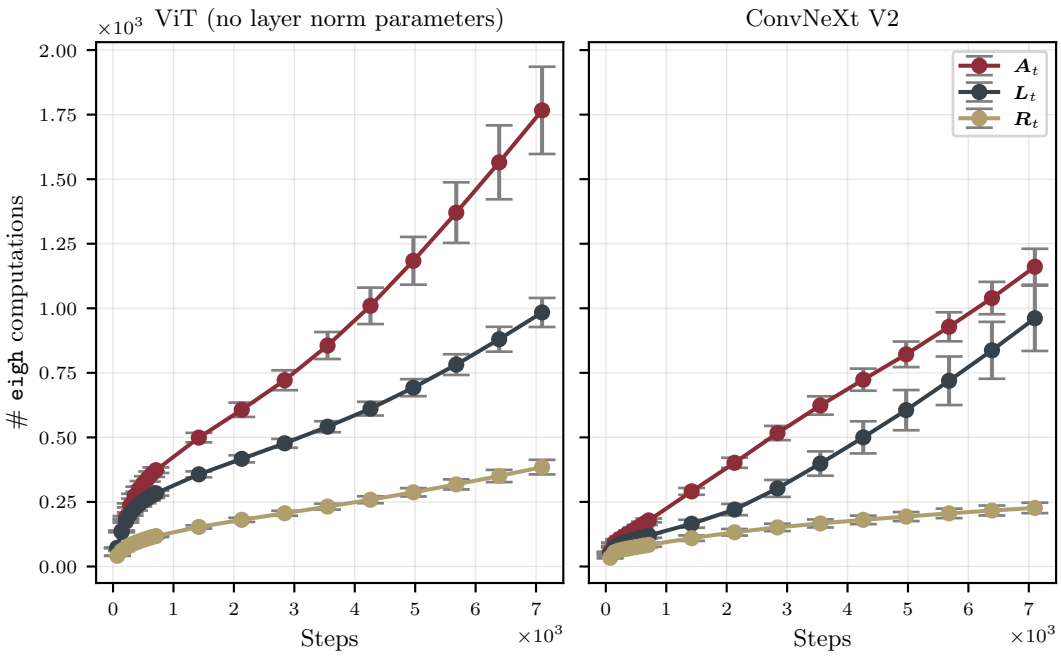

Figure 7: Same setting as in Figure 4. (**left**) When removing all learnable layer norm parameters from the ViT, the number of eigendecompositions for $\boldsymbol{L}_t$ and $\boldsymbol{R}_t$ remain approximately unchanged and no compensation appears to be happening. (**right**) With a ConvNeXt V2 architecture instead of the ViT, the overall pattern is similar, but with a more pronounced discrepancy between $\boldsymbol{L}_t$ and $\boldsymbol{R}_t$.

E.1 Imagewoof experimental details

We use the Imagewoof dataset.[13] All models are trained with cross entropy loss for 100 epochs, using a learning rate schedule consisting of a linear warmup for 353 steps followed by cosine decay. We use a batch size of 128, randomized cropping and horizontal flips as data augmentation, and the default settings for $\beta_1 = 0.9$ and $\beta_2 = 0.999$. For EShampoo in Figure 3, Figure 4, and Figure 5 we use the learning rate $\alpha = 6 \cdot 10^{-4}$ and $\epsilon = 10^{-10}$.

For the vision transformer (ViT) model, we use SimpleViT (Beyer et al., 2022) with patch size 16, 6 heads, a depth of 12 layers, an MLP dimension of 1536, dimension of 384, gradient clipping with threshold 1, and weight decay of 10^{-4}. For the ConvNeXt V2 architecture (Woo et al., 2023) we use weight decay of 0.05 and drop paths with rate 0.1.

For Figure 2, we fix $\epsilon = 10^{-10}$ and sweep the following learning rates α:[14]

- Vision transformer
  - AdamW: $\alpha \in \{10^{-4}, 3 \cdot 10^{-4}, 6 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-3}, 6 \cdot 10^{-3}, 10^{-2}\}$
  - Shampoo with grafting: $\alpha \in \{10^{-4}, 3 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-3}\}$
  - Shampoo: $\alpha \in \{3 \cdot 10^{-3}, 6 \cdot 10^{-3}, 10^{-2}, 3 \cdot 10^{-3}\}$
  - Shampoo^2 with trace scaling: $\alpha \in \{10^{-4}, 3 \cdot 10^{-4}, 6 \cdot 10^{-4}, 3 \cdot 10^{-3}\}$
  - EShampoo: $\alpha \in \{10^{-4}, 3 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-3}\}$
- ConvNeXt V2
  - AdamW: $\alpha \in \{10^{-4}, 3 \cdot 10^{-4}, 6 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-3}, 6 \cdot 10^{-3}, 10^{-2}\}$
  - Shampoo with grafting: $\alpha \in \{10^{-4}, 3 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-3}, 10^{-2}\}$
  - Shampoo: $\alpha \in \{10^{-4}, 3 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-3}\}$
  - Shampoo^2 with trace scaling: $\alpha \in \{10^{-4}, 3 \cdot 10^{-4}, 10^{-3}\}$
  - EShampoo: $\alpha \in \{10^{-4}, 3 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-3}, 10^{-2}\}$

[13] The Imagewoof dataset is available at https://github.com/fastai/imagenette.

[14] We fixed ϵ based on a wide sweep over α and ϵ for AdamW. We also use this ϵ when grafting from Adam.

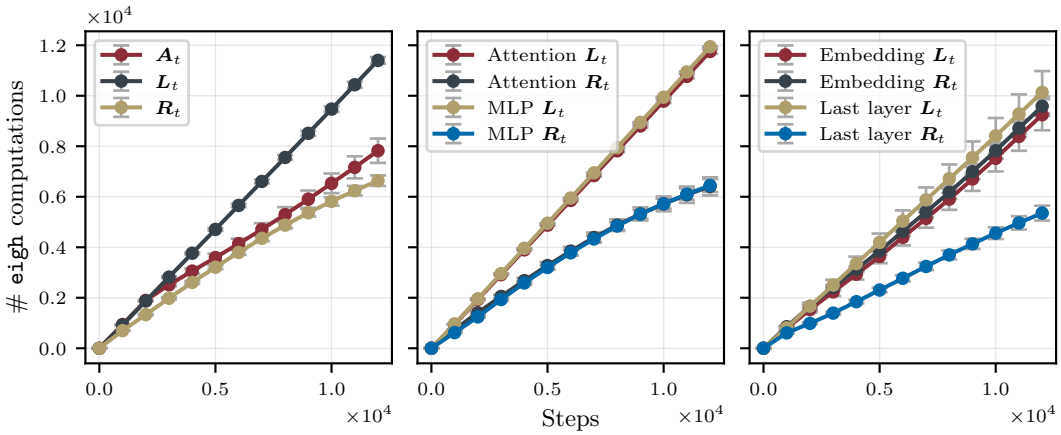

Figure 8: Llama 3 (324M) trained on 3.2B tokens of C4 data with EShampoo ($F = 1, \tau = 0.01$). The eigenbases for $\boldsymbol{L}_t$ are also in this setting consistently updated more frequently than for $\boldsymbol{R}_t$, with the exception of the embedding. The eigenbases of $\boldsymbol{A}_t$ are less frequently updated than of $\boldsymbol{L}_t$, in contrast to the Imagewoof experiments. There appears to be no difference in the pattern for weight matrices within the attention mechanisms and the MLPs.

E.2 More patterns of adaptivity

To test the generality of the trends described in Section 4.3, we perform the same experiment under a few additional settings.

In Figure 7, we remove all learnable layer norm parameters from the vision transformer and consider the ConvNeXt V2 model trained on the same dataset (Imagewoof). The overall pattern is consistent across all architectures. When removing the learnable layer norm parameters from the vision transformer, the number of eigendecompositions for $\boldsymbol{L}_t$ and $\boldsymbol{R}_t$ stays roughly constant (compare Figure 4 (right) with Figure 7 (left)). For the ConvNeXt V2 architecture, the discrepancy between the required updates for $\boldsymbol{L}_t$ and $\boldsymbol{R}_t$ is even more pronounced: on average, the eigenbases of $\boldsymbol{L}_t$ has to be updated significantly more frequently than for $\boldsymbol{R}_t$.

To expand beyond the vision modality and cover class imbalance, we also train a Llama 3 model with 324 million parameters on 3.2 billion tokens of the C4 dataset and conduct a similar analysis (Raffel et al., 2023; Grattafiori et al., 2024). We use EShampoo with $F = 1$ and $\tau = 0.01$.

In Figure 8 (left), we compute mean and standard errors across all (single) Kronecker factors $\boldsymbol{A}_t$ for RMS normalization parameters and all Kronecker factors $\boldsymbol{L}_t$ and $\boldsymbol{R}_t$ for linear layers and embedding blocks at every iteration. There are no bias parameters in this particular model. Consistent with the experiments in the vision setting, $\boldsymbol{L}_t$ is updated more frequently than $\boldsymbol{R}_t$, with $\boldsymbol{L}_t$ updated at almost every iteration, whereas $\boldsymbol{R}_t$ updated every other iteration. Unlike the vision setting, the Kronecker factors for the normalization layers $\boldsymbol{A}_t$ are initially updated at a similar frequency to $\boldsymbol{L}_t$, but diminishes after the first 25% of iterations until it is closer to, but still higher than, the update frequency for $\boldsymbol{R}_t$. The standard error for $\boldsymbol{A}_t$ is also larger than for $\boldsymbol{L}_t$ and $\boldsymbol{R}_t$.

In Figure 8 (middle), we consider two subsets of the hidden layers: the four weight matrices in the attention mechanism in each transformer block and the three weight matrices in the MLP in each transformer block. We compute the statistics across all weight matrices for each subset of the hidden weight matrices. There appears to be no significant difference in the number of eigendecompositions across steps between the two subsets of the weights.

In Figure 8 (right), we consider the input embedding layer and the last output layer. Due to the large vocabulary size, the gradients for these two layers are blocked such that no block has a dimension larger than 8192. Then we precondition each gradient block as usual with $\boldsymbol{L}_t$ and $\boldsymbol{R}_t$. Here, we compute the statistics across all of these blocks. The trend for the last layer is consistent with all other linear layers in our experiments. However, the eigendecomposition frequency of $\boldsymbol{L}_t$ and $\boldsymbol{R}_t$ for embeddings are almost identical.

Table 3: Results for all considered settings on a subset of the AlgoPerf workloads. We show the mean and standard error of the steps and time to the targets across the runs that actually hit the targets.

Workload	Shampoo Variant	Hits Targets	Steps	Time [min]
FastMRI	Adam grafting ($F = 100$)	$4/5$	4301 ± 109	13.96 ± 0.44
	$\boldsymbol{C}^{\text{EShampoo}}$ ($F = 100$)	$5/5$	2536 ± 66	10.44 ± 0.21
	$\boldsymbol{C}^{\text{EShampoo}}$ ($F = 50$)	$5/5$	2578 ± 86	10.86 ± 0.27
	$\boldsymbol{C}^{\text{EShampoo}}$ ($F = 10$)	$5/5$	2311 ± 73	14.93 ± 1.97
	$\boldsymbol{C}^{\text{EShampoo}}$ ($F = 1$)	$5/5$	2101 ± 31	35.34 ± 0.70
	$\boldsymbol{C}^{\text{EShampoo}}$ ($\tau = 0.1, F = 100$)	$5/5$	2553 ± 154	16.33 ± 2.10[17]
	$\boldsymbol{C}^{\text{EShampoo}}$ ($\tau = 0.1, F = 50$)	$5/5$	2468 ± 145	10.81 ± 0.72
	$\boldsymbol{C}^{\text{EShampoo}}$ ($\tau = 0.1, F = 10$)	$5/5$	2420 ± 92	10.76 ± 0.73
	$\boldsymbol{C}^{\text{EShampoo}}$ ($\tau = 0.1, F = 1$)	$5/5$	2367 ± 96	10.93 ± 0.42
	$\boldsymbol{C}^{\text{EShampoo}}$ ($\tau = 0.01, F = 1$)	$5/5$	2208 ± 41	27.43 ± 0.49
ImageNet ViT	Adam grafting ($F = 100$)	$1/1$	79907	894.27
	$\boldsymbol{C}^{\text{EShampoo}}$ ($F = 100$)	$1/1$	76226	894.85
	$\boldsymbol{C}^{\text{EShampoo}}$ ($F = 10$)	$1/1$	73237	1160.53
	$\boldsymbol{C}^{\text{EShampoo}}$ ($\tau = 0.1, F = 100$)	$1/1$	74010	852.66
	$\boldsymbol{C}^{\text{EShampoo}}$ ($\tau = 0.1, F = 50$)	$1/1$	77459	935.89
	$\boldsymbol{C}^{\text{EShampoo}}$ ($\tau = 0.01, F = 10$)	$1/1$	75841	1188.76
OGBG	Adam grafting ($F = 100$)	$2/5$	12574 ± 708	39.20 ± 1.88
	$\boldsymbol{C}^{\text{EShampoo}}$ ($F = 100$)	$3/5$	8320 ± 1203	33.02 ± 4.05
	$\boldsymbol{C}^{\text{EShampoo}}$ ($F = 50$)	$5/5$	7173 ± 443	26.17 ± 1.31
	$\boldsymbol{C}^{\text{EShampoo}}$ ($F = 10$)	$3/5$	6645 ± 357	37.55 ± 1.74
	$\boldsymbol{C}^{\text{EShampoo}}$ ($F = 1$)[18]	$-$	$-$	$-$
	$\boldsymbol{C}^{\text{EShampoo}}$ ($\tau = 0.1, F = 100$)	$4/5$	8047 ± 369	27.60 ± 1.15
	$\boldsymbol{C}^{\text{EShampoo}}$ ($\tau = 0.1, F = 50$)	$5/5$	7117 ± 328	27.55 ± 3.49
	$\boldsymbol{C}^{\text{EShampoo}}$ ($\tau = 0.1, F = 10$)	$5/5$	7151 ± 416	29.11 ± 1.98
	$\boldsymbol{C}^{\text{EShampoo}}$ ($\tau = 0.1, F = 1$)	$5/5$	6758 ± 273	34.16 ± 1.65
	$\boldsymbol{C}^{\text{EShampoo}}$ ($\tau = 0.01, F = 10$)	$2/5$	7234 ± 361	39.15 ± 2.01

E.3 AlgoPerf workloads

We follow the standard AlgoPerf setup and consider wall-clock time to pre-specified validation metric targets. See Dahl et al. (2023) and Kasimbeg et al. (2025) for more details on the AlgoPerf benchmark.[15] The FastMRI dataset can be attributed to Knoll et al. (2020); Zbontar et al. (2019), the ImageNet dataset to Krizhevsky et al. (2012), and the OGBG dataset to Hu et al. (2021).

We choose this specific subset of the workloads because (1) we want to include a larger scale vision transformer (ImageNet ViT), an architecture we use in the small-scale experiments, (2) the FastMRI and OGBG workloads share the same hyperparameter settings with the Imagenet ViT workload, hence excluding them as a confounding factor, and (3) the β_2 used for these workloads is the smallest among all hyperparameter settings of the winning Shampoo submission, resulting in the fastest moving average and thereby potentially faster changing eigenbases.

We run the winning Shampoo submission with Adam grafting, $F = 100$, and the best hyperparameter setting for each workload.[16] For EShampoo, we use the same best-performing hyperparameter setting from Shampoo but turn off learning rate grafting from Adam, and modify F and τ.

[15] The benchmark code is available at https://github.com/mlcommons/algorithmic-efficiency/.

[16] The submission is available at https://github.com/mlcommons/submissions_algorithms/tree/main/previous_leaderboards/algoperf_v05/submissions/external_tuning/shampoo_submission.

[17] This wall-clock time statistic seems to be negatively impacted by either an issue with the AlgoPerf code or the hardware setup.

[18] Runs failed due to https://github.com/mlcommons/algorithmic-efficiency/issues/866.

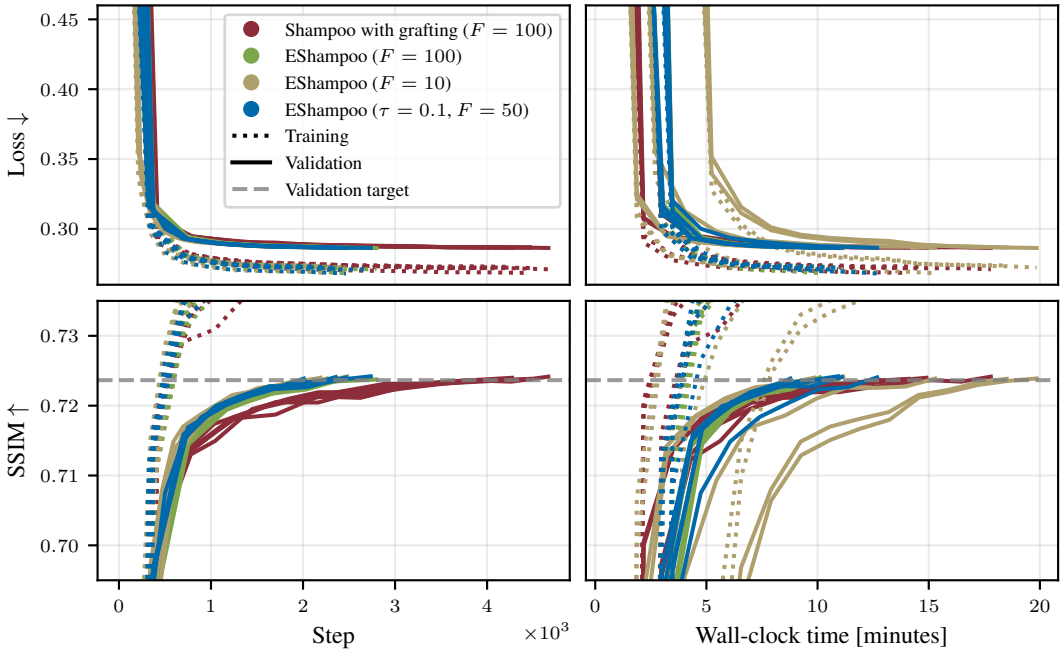

Figure 9: FastMRI AlgoPerf workload.

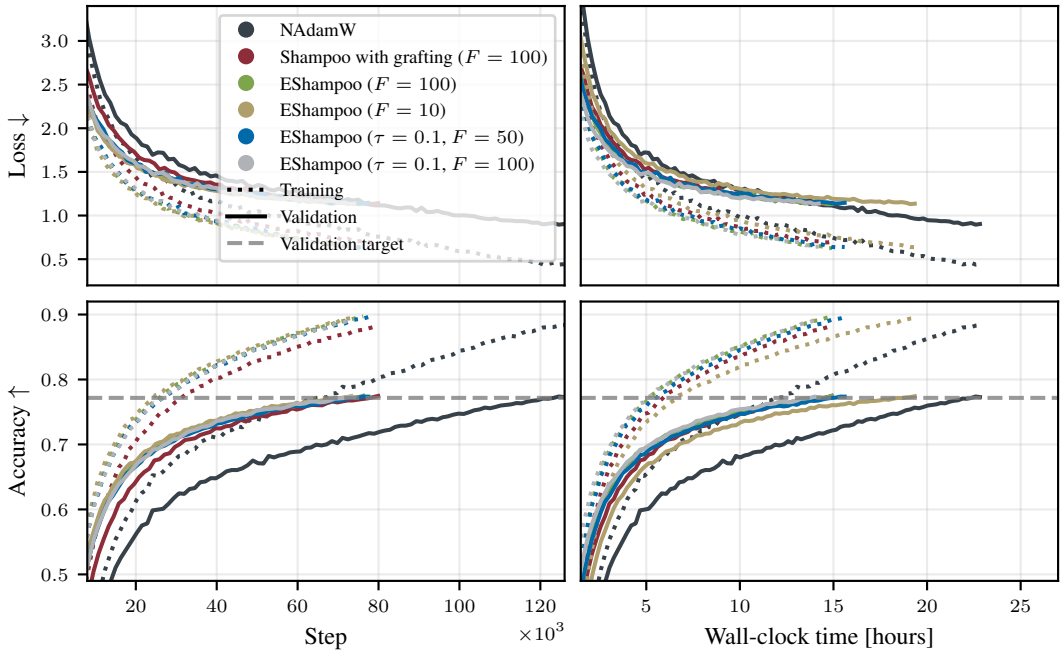

Figure 10: ImageNet ViT AlgoPerf workload.

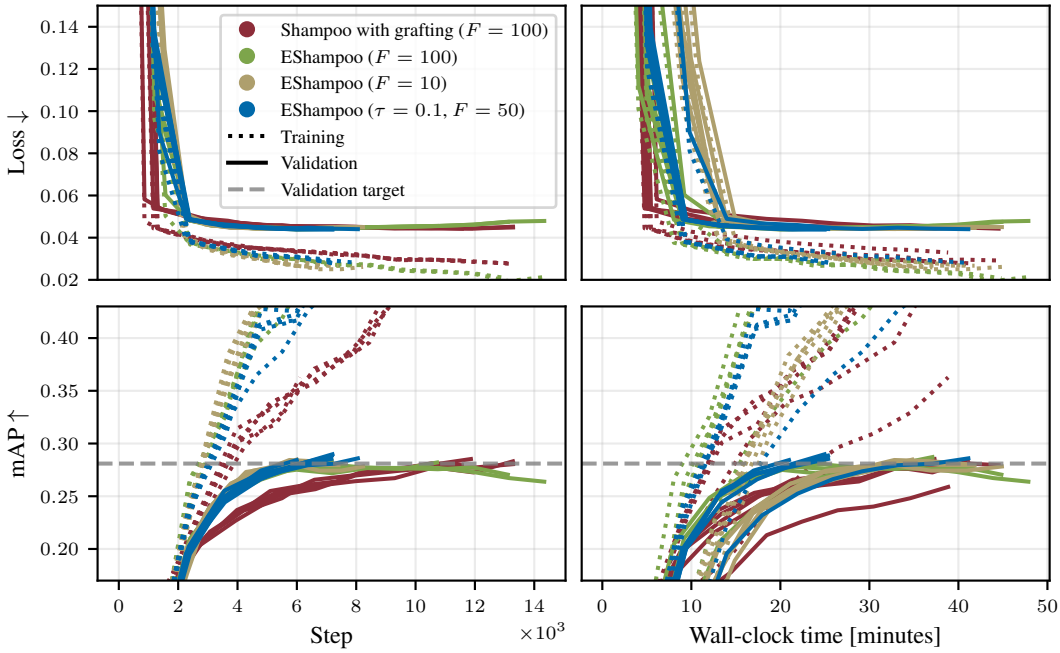

Figure 11: OGBG AlgoPerf workload.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Every claim is supported by empirical and theoretical evidence in Section 3 and Section 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We address limitations where appropriate and add a specific discussion of some important limiations in Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We provide proofs for all lemmas and propositions in Appendix C with an appropriate level of details and formality.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: While reproducing the exact numerical results will require the source code, we provide sufficient details to reproduce all the main claims of the paper in Section 3, Section 4, and Appendix E.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: All used datasets are open access. We do not provide the source code to reproduce all the experiments, but the implementation of all Shampoo variants is available at https://github.com/facebookresearch/optimizers and the AlgoPerf submission code for Shampoo is available at https://github.com/mlcommons/submissions_algorithms/tree/main/previous_leaderboards/algoperf_v05/submissions/external_tuning/shampoo_submission.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer:[Yes]

   Justification: The main section Section 3 and Section 4 provide all necessary details to understand the presented results and claims.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We do report standard errors of multiple runs for the AlgoPerf results in Table 1 and Table 3, with exception of the ImageNet ViT workload due to the larger computational cost. However, we do not report statistics over multiple runs for the sweeps presented Figure 1, Figure 2, and Figure 3 due to the large number of runs and the associated computation cost.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer:[Yes]

   Justification: We do provide details on the cluster used for the experiments and an estimate of the total amount of compute in Appendix E.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The paper covers fundamental algorithmic research and conforms with the all points of the NeurIPC Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: The work here is of fundamental nature and does not directly impact society. However, it might contribute to improved training algorithms which could potentially reduce the cost or improve the performance of arbitrary deep learning problems, including problems that impact society.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The work covers fundamental algorithmic research.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We do credit the creators of the datasets, model architectures, and the AlgoPerf benchmark in Appendix E.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA] .

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.