Can We Remove the Square-Root in Adaptive Gradient Methods? A Second-Order Perspective

Wu Lin [1] **Felix Dangel** [1] **Runa Eschenhagen** [2] **Juhan Bae** [3] **Richard E. Turner** [2] **Alireza Makhzani** [1 3]

Abstract

Adaptive gradient optimizers like Adam(W) are the default training algorithms for many deep learning architectures, such as transformers. Their diagonal preconditioner is based on the gradient outer product which is incorporated into the parameter update via a square root. While these methods are often motivated as approximate second-order methods, the square root represents a fundamental difference. In this work, we investigate how the behavior of adaptive methods changes when we remove the root, i.e. strengthen their second-order motivation. Surprisingly, we find that such square-root-free adaptive methods *close* the generalization gap to SGD on convolutional architectures, while *maintaining* their root-based counterpart's performance on transformers. The second-order perspective also has practical benefits for the development of non-diagonal adaptive methods through the concept of *preconditioner invariance*. In contrast to root-based methods like Shampoo, the root-free counterparts do not require numerically unstable matrix root decompositions and inversions, thus work well in half precision. Our findings provide new insights into the development of adaptive methods and raise important questions regarding the currently overlooked role of adaptivity for their success.

1. Introduction

Adaptive gradient-based methods like Adam (Kingma & Ba, 2015) play a significant role in training modern deep learning models such as transformers. A better understanding of these methods allows us to address their shortcomings and develop new adaptive methods to reduce training time and improve performance. This is essential as deep learning models become increasingly large and complex to train.

[1]Vector Institute, Canada [2]Cambridge University, United Kingdom [3]University of Toronto, Canada. Correspondence to: Wu Lin <yorker.lin@gmail.com >.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

Despite their success on architectures like transformers, adaptive methods tend to generalize worse than stochastic gradient descent (SGD) on convolutional architectures (Wilson et al., 2017). Our understanding of this discrepancy is limited. Balles & Hennig (2018) dissect Adam into two concepts, sign descent and adaptive step sizes, and hypothesize that the connection to sign descent causes the gap with SGD on CNNs. Similarly, Kunstner et al. (2023); Chen et al. (2023) argue that adaptive methods outperform SGD on transformers due to their connection to sign descent.

It is challenging to isolate the sign descent component of adaptive methods like Adam or RMSProp (Tieleman & Hinton, 2012), as they typically introduce a square root to the preconditioner, which conflates the sign and adaptivity aspects and hinders our comprehension of the role of the adaptivity. The root is motivated by reports of improved performance (Tieleman & Hinton, 2012) and to stabilize convergence near the optimum (Kingma & Ba, 2015; Kunstner et al., 2019; Martens, 2020). However, it conflicts with the motivation of adaptive methods as approximate second-order methods based on the empirical Fisher, which is also commonly mentioned in works introducing adaptive methods (e.g. Kingma & Ba, 2015).

Here, we investigate how the behavior of adaptive methods changes when we remove the root. Our idea is to strengthen the often-mentioned link to second-order methods that is weakened by the root. Conceptually, this cleanly disentangles the aforementioned adaptivity aspect from the sign aspect. Practically, it provides an opportunity to revisit the root's role in the context of modern training strategies, like non-constant learning rate schedules (Loshchilov & Hutter, 2016) and hyperparameter tuning schemes (Choi et al., 2019), that differ from the original pipelines in which the square root was introduced. Computationally, removing the root is beneficial for bridging the computation gap (Figure 3) between diagonal and non-diagonal *matrix* adaptive methods and lowering the per-iteration cost of matrix methods by removing matrix root decompositions that need to be carried out in high precision to avoid numerical instabilities (Gupta et al., 2018; Anil et al., 2020).

There are some challenges to establishing a rigorous second-order perspective on adaptive methods; one cannot just remove the root. We overcome those challenges and make the

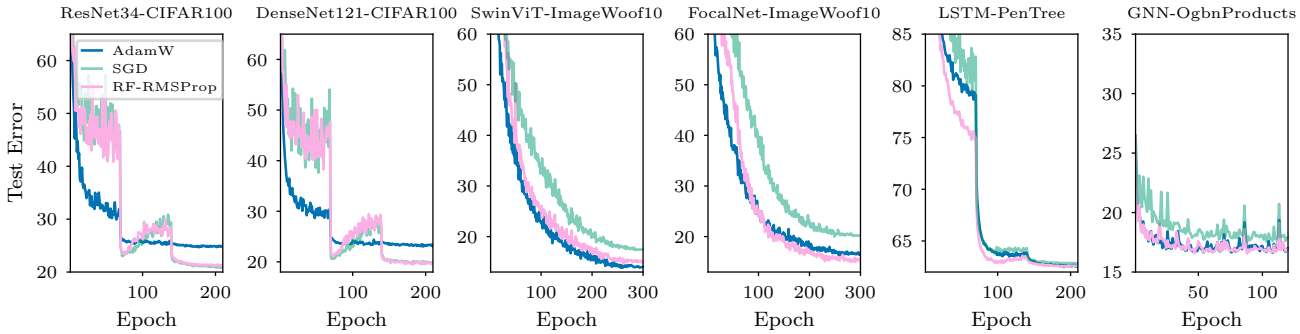

Figure 1. In modern (pre-)training setups (learning rate schedule, random search using 200 runs), square-root-free (RF) adaptive methods close the generalization gap between their root-based counterparts and SGD on CNNs (CIFAR-100), while maintaining their performance on vision transformers (ImageWoof10). They work well on other problems, like training a 3-layer LSTM, and a GNN with attention (Zhang et al., 2022). Experimental setup, performance measurements, and fine-tuning experiments on vision models are described in Appendix J.

following contributions:

- We establish a rigorous second-order view of adaptive methods: we remove the root (Sec. 2), show how to interpret the gradient outer product as a *new* empirical Fisher variant (Sec. 3), adjust the preconditioner initialization (Sec. 4), and emphasize the importance of incorporating mini-batch curvature approximations (e.g., the outer products) from previous iterations (Secs. 2,3,4).
- Empirically, we show that—surprisingly—removing the root not only *closes* the generalization gap between adaptive methods and SGD on convolutional neural networks, but *maintains* the performance of square-root-based methods on vision transformers (Sec. 2).
- Conceptually, we introduce *preconditioner invariance* (Sec. 4) through the second-order view to incorporate arbitrary curvature approximations, remove inverses in root-free matrix adaptive methods, and bridge the computation gap between diagonal and matrix adaptive methods by developing new inverse- and root-free matrix methods that can train in low precision (Fig. 3).
- Consequently, we propose root-free RMSProp (Fig. 4) and root- and inverse-free Shampoo (Fig. 5) which are invariant to scaling the loss and affine reparametrization (Sec. 3), and work well on a variety of models (CNNs, LSTMs, GNNs, ViTs, and VMambas).

2. First-order View of Adaptive Methods

For many deep learning tasks, training a neural network (NN) means solving an unconstrained optimization problem. For simplicity, consider a supervised learning task with a set of N data points $\{y_i, \mathbf{x}_i\}_{i=1}^N$, where y_i and $\mathbf{x}_i$ represent a label and a feature vector. Given a NN $f(\cdot; \boldsymbol{\mu})$ with learnable weights $\boldsymbol{\mu}$, the optimization problem is

$$\min_{\mu} \ell(\boldsymbol{\mu}) \coloneqq \textstyle\sum_{i=1}^N c(y_i, f(\mathbf{x}_i; \boldsymbol{\mu}))\,, \tag{1}$$

where $\hat{y}_i \coloneqq f(\mathbf{x}_i; \boldsymbol{\mu})$ is a predicted label of a feature vector $\mathbf{x}_i$ and $c(y_i, \hat{y}_i)$ is a loss function that measures the discrepancy between a true (y_i) and predicted ($\hat{y}_i$) label.

To minimize (1), we can use adaptive gradient methods, which use the following preconditioned gradient update

$$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \beta_1 \mathbf{S}^{-1} \nabla_{\mu} \ell(\boldsymbol{\mu})\,, \tag{2}$$

where β_1 is an (initial) learning rate, $\nabla_{\mu}\ell(\boldsymbol{\mu})$ is a gradient vector, and $\mathbf{S}$ is a preconditioning matrix. When $\mathbf{S}$ is the Hessian and $\beta_1 = 1$, this becomes Newton's method. In adaptive gradient methods, the preconditioning matrix $\mathbf{S}$ is estimated by using only gradient information.

To estimate the preconditioner $\mathbf{S}$, many adaptive methods employ an outer product $\mathbf{H} \coloneqq \mathbf{g}\mathbf{g}^\top$ of a gradient vector $\mathbf{g}$ that is usually the mini-batch gradient of Equation (1): *Diagonal* adaptive methods such as RMSProp (Tieleman & Hinton, 2012), diag-AdaGrad (Duchi et al., 2011), and Adam (Kingma & Ba, 2015), only use the diagonal entries $\operatorname{diag}(\mathbf{H}) = \mathbf{g} \odot \mathbf{g}$ where $\odot$ denotes element-wise multiplication. *Full matrix* adaptive methods like full-AdaGrad (Duchi et al., 2011) and its variants (Agarwal et al., 2019) use the gradient outer product $\mathbf{H}$. *Structured matrix* methods like Shampoo (Gupta et al., 2018) use a Kronecker-factored approximation of the gradient outer product $\mathbf{H}$.

All of these methods apply a square root to the preconditioner in their update, and we will refer to them as square-root-based methods. For example, RMSProp updates its preconditioner $\mathbf{S} = \operatorname{diag}(\sqrt{\hat{\mathbf{s}}})$ and parameters according to

$$\hat{\mathbf{s}} \leftarrow (1-\beta_2)\hat{\mathbf{s}} + \beta_2 \mathbf{h}\,,\ \boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \beta_1 \mathbf{S}^{-1}\mathbf{g} = \boldsymbol{\mu} - \beta_1 \frac{\mathbf{g}}{\sqrt{\hat{\mathbf{s}}}}\,, \tag{3}$$

where $\mathbf{h} \coloneqq \operatorname{diag}(\mathbf{H})$, β_1 is the learning rate, and β_2 is the weight to incorporate new outer products. In full-matrix cases, adding the root requires matrix root decomposition. For example, full-AdaGrad computes the matrix root for its preconditioner $\mathbf{S} = \hat{\mathbf{S}}^{1/2}$ and updates parameters as

$$\hat{\mathbf{S}} \leftarrow \hat{\mathbf{S}} + \beta_2 \mathbf{g}\mathbf{g}^\top\,,\ \boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \beta_1 \mathbf{S}^{-1}\mathbf{g}\,. \tag{4}$$

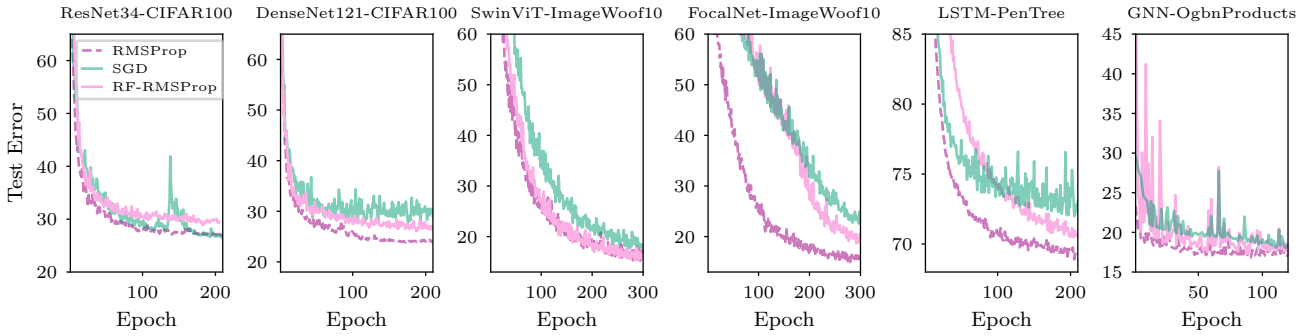

Figure 2. Comparison of root-based versus square-root-free (RF) methods in the original (outdated) training setup the root was introduced in (see Appendix J for details). Adaptive methods with the root work better than their root-free counterparts when using (1) a constant learning rate schedule, (2) default zero initialization for the preconditioner, (3) default scaling for an averaged mini-batch loss.

2.1. Benefits and Results of Removing the Square Root

It is unclear how *exactly* the square root emerged in adaptive methods. Here, we hypothesize how it got introduced, critically assess why it may be desirable to remove, and highlight connections to our experimental results. We find that few works study square-root-free adaptive methods, and that benefits of removing the root are often overlooked. Our work fills in this gap, which we think is crucial to better understand these methods and our empirical results underline the great potential of square-root-free adaptive methods.

Empirical performance & training schemes The most important motivation in favor of the square root is the strong empirical performance of square-root-based methods that has been demonstrated in various works and rightfully established them as state-of-the-art for training transformers. However, the context in which the root was introduced significantly differed from the training schemes that are used nowadays. Original works such as Tieleman & Hinton (2012) show that including the square root improves the performance of adaptive methods when only tuning a learning rate that is fixed throughout training (Bottou et al., 2018). Beyond this original setting, square-root-based methods have demonstrated great capabilities to train a wide range of NNs, e.g. when using a non-constant learning rate (Loshchilov & Hutter, 2016) and random search (Bergstra & Bengio, 2012; Choi et al., 2019) to tune all available hyperparameters. We wonder whether the square root, while necessary to achieve good performance in outdated training schemes, might *not* be required in contemporary schemes. To investigate this hypothesis, we conducted an experiment that compares square-root-based and square-root-free methods using the original training scheme in which the square root was introduced (Figure 2). Indeed, we find that the square root is beneficial in the original context.

Interpretability & generalization Square-root-based methods are state-of-the-art for training attention-based models (Zhang et al., 2020), but exhibit a generalization gap with SGD on CNNs (Wilson et al., 2017). The square root complicates the understanding of this phenomenon since adding the root conflates sign descent and adaptivity. The sign-based component is often considered as a salient feature of adaptive methods while adaptivity is not. For example, recent studies attribute the superior performance of adaptive methods over SGD on attention models to their sign-based update rules (Kunstner et al., 2023; Chen et al., 2023). On the other hand, Balles & Hennig (2018) hypothesize that sign descent may cause poor generalization of Adam on CNNs. However, they neither consider the direct effect of the square root nor remove it from an existing method. Therefore, the role of adaptivity is overlooked when it comes to understanding the gap between adaptive methods and SGD. Removing the square root could clarify this issue, as a root-free method no longer performs sign descent while preserving adaptivity. To investigate the role of adaptivity, we experiment with root-free RMSProp (Figure 4) on attention and convolutional models (Figure 1). For attention architectures, we find that removing the root does *not* harm the performance of RMSProp. Root-free methods with update clipping (Liu et al., 2023) also perform well on large transformer-based language models. These findings suggest that adaptivity is equally important as sign descent to explain the performance gap between adaptive methods and SGD on transformers. For convolutional architectures, removing the root closes the generalization gap between RMSProp and SGD. This suggests that root-free adaptive methods can generalize well, and raises novel questions on the understanding of the role of adaptivity.

Computational cost As noted by Duchi et al. (2011), the root poses numerical and computational challenges on *matrix* methods like Eq. (4) as it requires matrix roots which must be carried out in high precision to avoid numerical instabilities. This increases the run time and memory footprint and complicates the implementation (Anil et al., 2020; Shi et al., 2023). Using low-precision data types (Micikevicius et al., 2017) is a key technique to boost training speed and reduce memory. The square root makes matrix methods undesirable for modern low-precision training schemes even when we can use iterative solvers (Anil et al., 2020) to compute matrix roots. By removing the matrix roots,

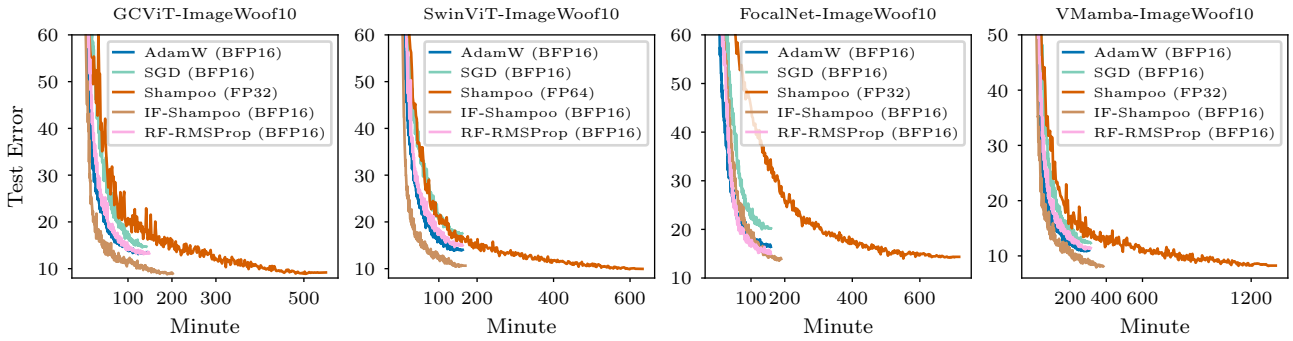

Figure 3. Comparison of matrix root-free versus root-based methods on GCViT (Hatamizadeh et al., 2023), SwinViT (Liu et al., 2021), FocalNet (Yang et al., 2022), and VMamba (Liu et al., 2024). Both matrix methods (Shampoo, IF-Shampoo) outperform diagonal methods on modern vision models such as transformers using modern training strategies. In contrast to Shampoo, our inverse-free matrix method, IF-Shampoo, runs in BFP-16 and trains twice as fast, while using less memory. All models are trained for 300 epochs. We update matrix preconditioners at every 2 iterations and can reduce the clock time by updating them less frequently. See App. J for more details.

and using advances on second-order methods (Lin et al., 2023b), we develop inverse-free matrix methods (Sec. 4.3) that are suitable for mixed-precision training. We present empirical results for the low-precision setting in Fig. 3. By removing the roots, we can consistently train in BFP-16, whereas root-based matrix methods like Shampoo need single—sometimes even double—precision. On modern vision models, we find that root-free methods outperform diagonal methods and perform similarly to Shampoo while requiring less time and memory (see Fig. 3) due to low-precision training. This shows that removing the roots allows us to overcome the challenges of existing matrix methods, bridge the computation gap with diagonal methods, and expand their applicability to modern training pipelines.

Theoretical considerations *Invariances.* Adding the root makes a descent step invariant to the scale of the loss – the square root adjusts the scale of the "squared" gradient to be consistent with the gradient when the loss function is scaled. This is useful as there is no need for users to pay attention to whether the loss function is averaged or summed over data points. While adding the square root fixes the scaling issue, it breaks the affine reparameterization invariance (Nesterov & Nemirovskii, 1994). We can fix the scaling issue and preserve the affine invariance without the square root, as will be shown in Section 3; see Appendix A for an example of the affine invariance of square-root-free methods. Moreover, removing the root makes it easy to introduce a new kind of invariance for preconditioning matrices (Section 4).

Convergence analysis. Theoretical works such as Duchi et al. (2011); Reddi et al. (2019) and many others suggest that adding the square root is useful to prove convergence bounds for convex loss functions. Convergence analysis for square-root-based methods is also extended to non-convex settings under certain assumptions such as gradient Lipschitz and Polyak-Łojasiewicz conditions. However, compared to the regret bound of AdaGrad (Duchi et al., 2011) studied in convex settings, Hazan et al. (2006) give a better regret bound in strongly-convex cases *without* introducing the square root. Recent works like Mukkamala & Hein (2017); Wang et al. (2020) prove similar bounds for square-root-free methods on convex problems. Thus, square-root-free methods are theoretically grounded—at least in convex settings.

Behavior near optimum. The square root is often introduced to avoid oscillation near an optimal solution where the preconditioner can be ill-conditioned. For example, in one dimension, the descent direction $s^{-1}g$ can be unbounded near the optimum when we use the outer product as the preconditioner $s = g^2$. This is because the gradient g is close to 0 near the optimum and the outer product as a 'squared' gradient decreases much faster than the gradient g. Thus, the update without the square root can lead to oscillation when using a constant learning rate. However, even without the square root, a preconditioner can still be well-conditioned when incorporating outer products from previous iterations (Roux et al., 2007) and using Tikhonov damping (Becker et al., 1988; Martens & Sutskever, 2012). For example, the descent direction $s^{-1}g$ can be bounded without the square root even when near an optimal solution, where g is a gradient and $s = (1-\beta_2)s + \beta_2 g^2$ is a preconditioner estimated by an exponentially moving average when $0 < \beta_2 < 1$.

3. A Second-order Perspective

Here, we describe a second-order perspective on adaptive methods when the root is removed. This naturally resolves the scaling issue and is coherent with the original motivation to develop these methods.

Without the square root, an adaptive method takes a step

$$\mathbf{S} \leftarrow (1-\beta_2\gamma)\mathbf{S} + \beta_2\mathbf{H}\,, \qquad \boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \beta_1\mathbf{S}^{-1}\mathbf{g}\,, \tag{5}$$

where $\gamma \in \{0, 1\}$ and the outer product $\mathbf{H} = \mathbf{g}\mathbf{g}^\top$ is used as a curvature approximation. Note that we incorporate outer products from previous iterations as $\beta_2 > 0$. For example, when $\gamma = 0$ and $\mathbf{S}$ is a full matrix, we obtain the update of full-AdaGrad without the square root.

The above update resembles a second-order method. How-

RMSProp

1: Compute gradient $\mathbf{g} \coloneqq \nabla \ell_{\text{scaled}}(\boldsymbol{\mu})$
$\hat{\mathbf{s}} \leftarrow (1-\beta_2)\hat{\mathbf{s}} + \beta_2 \mathbf{g}^2$
2: $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \beta_1 \mathbf{g}/\sqrt{\hat{\mathbf{s}}}$

Square-root-free RMSProp (Ours)

1: Compute gradient $\mathbf{g} \coloneqq \nabla \ell_{\text{scaled}}(\boldsymbol{\mu})$
$\mathbf{s} \leftarrow (1-\beta_2)\mathbf{s} + \beta_2 \mathbf{B}\mathbf{g}^2$
2: $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \beta_1 \mathbf{g}/\mathbf{s}$

Figure 4. Diagonal adaptive methods for a (scaled) loss function $\ell_{\text{scaled}}(\boldsymbol{\mu})$ defined by averaging over B data points in a mini-batch case. We initialize s to 1 in the square-free method in our square-root-free method while $\hat{s}$ is initialized to 0 in the original RMSProp. For simplicity, we do not include damping, weight decay, and momentum. A full-fledged version can be found in the Appendix, Figure 6.

ever, it is not invariant to the scale of the loss, unlike Newton's method: Scaling the loss by a constant c scales the gradient and Hessian by c, but the gradient outer product by c^2, which is inconsistent with its role as a Hessian approximation and therefore conflicts with the second-order interpretation. We will resolve this particular conflict and improve the justification of square-root-free adaptive methods through an approximate second-order view.

The key step is to view the outer product as a novel empirical Fisher that *differs* from the standard empirical Fisher discussed in the DL literature (Kingma & Ba, 2015; Kunstner et al., 2019; Martens, 2020). Our empirical Fisher relies on the aggregated mini-batch gradient rather than per-sample gradients. Since the Fisher is tied to a probability distribution that must be normalized, this provides a gauge to make updates invariant to the scale of the loss automatically.

3.1. New Fisher Matrices as Hessian Approximations

Recall the objective function $\ell(\boldsymbol{\mu}) = \sum_{i=1}^N c(y_i, f(\mathbf{x}_i; \boldsymbol{\mu}))$ defined in (1). Given a datum $\mathbf{x}_i$, we can define a per-example probability distribution over label y_i as $p(y_i \mid \mathbf{x}_i; \boldsymbol{\mu}) = \exp(-c(y_i, f(\mathbf{x}_i; \boldsymbol{\mu})))$ by using the loss function $c(y_i, f(\mathbf{x}_i; \boldsymbol{\mu}))$ (e.g., cross-entropy or square loss). We denote an individual gradient for datum $\mathbf{x}_i$ by $\mathbf{g}_i \coloneqq \nabla_\mu c(y_i, f(\mathbf{x}_i; \boldsymbol{\mu})) = -\nabla_\mu \log p(y_i | \mathbf{x}_i; \boldsymbol{\mu})$.

(1) Standard empirical Fisher matrices

Definition 1. The empirical Fisher information (FIM) matrix (Pascanu & Bengio, 2013; Kingma & Ba, 2015; Kunstner et al., 2019; Martens, 2020) $\tilde{\mathbf{F}}_{\text{standard}}(\boldsymbol{\mu})$ is defined by replacing the expectation in the standard FIM $\mathbf{F}_{\text{standard}}(\boldsymbol{\mu})$ with observed labels.

$$\mathbf{F}_{\text{standard}}(\boldsymbol{\mu}) \coloneqq \sum_{i=1}^N \mathbb{E}_{y_i \sim p(y_i|\mathbf{x}_i;\boldsymbol{\mu})}\left[\nabla_\mu \log p(y_i \mid \mathbf{x}_i; \boldsymbol{\mu}) \nabla_\mu^\top \log p(y_i \mid \mathbf{x}_i; \boldsymbol{\mu})\right]$$

$$\tilde{\mathbf{F}}_{\text{standard}}(\boldsymbol{\mu}) \coloneqq \sum_{i=1}^N \nabla_\mu \log p(y_i \mid \mathbf{x}_i; \boldsymbol{\mu}) \nabla_\mu^\top \log p(y_i \mid \mathbf{x}_i; \boldsymbol{\mu}) = \sum_{i=1}^N \mathbf{g}_i \mathbf{g}_i^\top$$

Now, we introduce a new Fisher matrix as a FIM over a *joint* distribution of labels.

(2) Our Fisher matrices for the original (unscaled) loss

Definition 2. Our Fisher matrix is defined as

$$\begin{aligned}\mathbf{F}_{\text{new}}(\boldsymbol{\mu}) &\coloneqq \mathbb{E}_{\mathbf{y} \sim p(\mathbf{y}|\mathbf{X};\boldsymbol{\mu})}\left[\nabla_\mu \log p(\mathbf{y} \mid \mathbf{X}; \boldsymbol{\mu}) \nabla_\mu^\top \log p(\mathbf{y} \mid \mathbf{X}; \boldsymbol{\mu})\right] \\ &= -\mathbb{E}_{\mathbf{y} \sim p(\mathbf{y}|\mathbf{X};\boldsymbol{\mu})}\left[\nabla_\mu^2 \log p(\mathbf{y} \mid \mathbf{X}; \boldsymbol{\mu})\right],\end{aligned} \tag{6}$$

where the labels $\mathbf{y} = (y_1, \ldots, y_N)$ are considered jointly as a random vector, $p(\mathbf{y}|\mathbf{X}; \boldsymbol{\mu}) \coloneqq \prod_{i=1}^N p(y_i \mid \mathbf{x}_i; \boldsymbol{\mu})$ is its joint distribution, and $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ is a feature matrix. Importantly, the joint distribution must be normalized.

Definition 3. Our empirical Fisher matrix is defined by replacing the expectation in $\mathbf{F}_{\text{new}}(\boldsymbol{\mu})$ with observed labels.

$$\begin{aligned}\tilde{\mathbf{F}}_{\text{new}}(\boldsymbol{\mu}) &\coloneqq \nabla_\mu \log p(\mathbf{y} \mid \mathbf{X}; \boldsymbol{\mu}) \nabla_\mu^\top \log p(\mathbf{y} \mid \mathbf{X}; \boldsymbol{\mu}) = \mathbf{H} \\ &\approx -\nabla_\mu^2 \log p(\mathbf{y} \mid \mathbf{X}; \boldsymbol{\mu}) = \nabla_\mu^2 \ell(\boldsymbol{\mu}) \,,\end{aligned} \tag{7}$$

with $\nabla_\mu \log p(\mathbf{y} \mid \mathbf{X}; \boldsymbol{\mu}) = \sum_i \nabla_\mu \log p(y_i \mid \mathbf{x}_i; \boldsymbol{\mu}) = -\sum_i \mathbf{g}_i = -\mathbf{g}$.

From (7), we see that the outer product $\mathbf{H} = \mathbf{g}\mathbf{g}^\top$ coincides with this empirical Fisher matrix $\tilde{\mathbf{F}}_{\text{new}}(\boldsymbol{\mu})$.

Definition 4. In a mini-batch case, we can define a Fisher matrix for a mini-batch with B data points as

$$\mathbf{F}_{\text{mini}}(\boldsymbol{\mu}) \coloneqq \mathbb{E}_{\mathbf{y}_{\text{mini}} \sim p}\left[\nabla_\mu \log p(\mathbf{y}_{\text{mini}} \mid \mathbf{X}_{\text{mini}}; \boldsymbol{\mu}) \nabla_\mu^\top \log p(\mathbf{y}_{\text{mini}} \mid \mathbf{X}_{\text{mini}}; \boldsymbol{\mu})\right] \tag{8}$$

where $\mathbf{y}_{\text{mini}} \coloneqq (y_1, \ldots, y_B)$ is a label vector, $\mathbf{X}_{\text{mini}} \coloneqq (\mathbf{x}_1, \ldots, \mathbf{x}_B)$ a feature matrix for the mini-batch, and $p(\mathbf{y}_{\text{mini}} \mid \mathbf{X}_{\text{mini}}; \boldsymbol{\mu}) \coloneqq \prod_{i=1}^B p(y_i \mid \mathbf{x}_i, \boldsymbol{\mu})$ the joint distribution over labels for the mini-batch. This distribution can also be obtained by marginalizing unseen labels in the original joint distribution $p(\mathbf{y} \mid \mathbf{X}; \boldsymbol{\mu})$ defined for the full-batch. This Fisher is an unbiased estimation of our full-batch Fisher as the claim—proof in Appendix B—is stated below.

Claim 1. Our mini-batch Fisher $1/B \mathbf{F}_{\text{mini}}(\boldsymbol{\mu})$ is an unbiased estimation of the full-batch Fisher $1/N \mathbf{F}_{\text{new}}(\boldsymbol{\mu})$.

When we replace the expectation with observed labels, we obtain our empirical Fisher for the mini-batch. Note that this empirical Fisher is not an unbiased estimator of the full-batch empirical Fisher. However, we often do not consider the unbiasedness when we incorporate empirical Fisher matrices from previous iterations. Moreover, our interpretation provides the direct link to the Hessian as we can view the outer product as a Hessian approximation (see (7)) when the product corresponds to our Fisher. This interpretation allows us to preserve the scale-invariance to the loss and the affine reparametrization invariance.

(3) Our Fisher matrices for a scaled loss

Now, we consider a scaled loss function. Then, the outer product $\mathbf{H}$ no longer coincides with a Fisher matrix. As an example, we consider averaging a loss over $N > 1$ data points, as it is often done in mini-batch settings. The loss is

$$\ell_{\text{scaled}}(\boldsymbol{\mu}) \coloneqq \tfrac{1}{N}\ell(\boldsymbol{\mu}) = \tfrac{1}{N}\sum_{i=1}^N c(y_i, f(\mathbf{x}_i; \boldsymbol{\mu})),$$

with gradient $\mathbf{g}_{\text{scaled}} \coloneqq {}^1\!/_N \sum_{i=1}^N \nabla_\mu c(y_i, f(\mathbf{x}_i;\boldsymbol{\mu})) = {}^1\!/_N \sum_{i=1}^N \mathbf{g}_i$ and outer product and $\mathbf{H}_{\text{scaled}} \coloneqq \mathbf{g}_{\text{scaled}}\mathbf{g}_{\text{scaled}}^\top = {}^1\!/_{N^2}\mathbf{H}$.

In this case, the empirical Fisher matrix should be defined using the same joint distribution as

$$\begin{aligned}\tilde{\mathbf{F}}_{\text{scaled}}(\boldsymbol{\mu}) &\coloneqq \frac{1}{N}\nabla_\mu \log p(\mathbf{y} \mid \mathbf{X};\boldsymbol{\mu})\nabla_\mu^\top \log p(\mathbf{y} \mid \mathbf{X};\boldsymbol{\mu}) = N\mathbf{H}_{\text{scaled}} \\ &= \frac{1}{N}\tilde{\mathbf{F}}_{\text{new}}(\boldsymbol{\mu}) \approx -\frac{1}{N}\nabla_\mu^2 \log p(\mathbf{y} \mid \mathbf{X};\boldsymbol{\mu}) = \nabla_\mu^2 \ell_{\text{scaled}}(\boldsymbol{\mu})\,.\end{aligned} \tag{9}$$

According to (9), the outer product $\mathbf{H}_{\text{scaled}}$ no longer coincides with this empirical Fisher matrix $\tilde{\mathbf{F}}_{\text{scaled}}(\boldsymbol{\mu})$. When using $\tilde{\mathbf{F}}_{\text{scaled}}(\boldsymbol{\mu})$ as the Hessian approximation, the update of a square-root-free adaptive method is

$$\mathbf{S} \leftarrow (1-\beta_2\gamma)\mathbf{S} + \beta_2 \overbrace{\tilde{\mathbf{F}}_{\text{scaled}}(\boldsymbol{\mu})}^{=N\mathbf{H}_{\text{scaled}}}, \quad \boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \beta_1 \mathbf{S}^{-1}\mathbf{g}_{\text{scaled}}$$

This update is scale-invariant since gradient $\mathbf{g}_{\text{scaled}} = {}^1\!/_N\mathbf{g}$ and empirical Fisher $\tilde{\mathbf{F}}_{\text{scaled}}(\boldsymbol{\mu}) = N\mathbf{H}_{\text{scaled}} = {}^1\!/_N\mathbf{H}$ scale identically.

Affine reparametrization invariance Our fix does not break the affine invariance of a square-root-free method. A square-root-free method is affine invariant when using our scaled empirical Fisher matrix (see Appx. C for a proof).

Claim 2. Our square-root-free update is affine invariant.

By using our empirical Fisher matrices, our square-root-free methods like Newton's method not only is scale-invariant when we scale a loss function but also preserves the affine reparametrization invariance. Thus, we consider our methods as approximate second-order (quasi-Newton) methods.

3.2. Difference to the Standard Empirical Fisher

Existing works (Kingma & Ba, 2015; Kunstner et al., 2019; Martens, 2020) do not distinguish the standard empirical Fisher denoted by $\tilde{\mathbf{F}}_{\text{standard}}(\boldsymbol{\mu})$ from the outer product $\mathbf{H}$ corresponding to our empirical Fisher $\tilde{\mathbf{F}}_{\text{new}}(\boldsymbol{\mu})$. They differ when $N > 1$ since

$$\tilde{\mathbf{F}}_{\text{standard}}(\boldsymbol{\mu}) = \sum_{i=1}^N (\mathbf{g}_i\mathbf{g}_i^\top) \neq \left(\sum_{i=1}^N \mathbf{g}_i\right)\left(\sum_{j=1}^N \mathbf{g}_j\right)^\top = \mathbf{H} = \tilde{\mathbf{F}}_{\text{new}}(\boldsymbol{\mu})\,. \tag{10}$$

The standard empirical Fisher on the left is not rank-one, while our empirical Fisher on the right is.

This misconception can impede our understanding of these methods. The outer product $\mathbf{H}$ corresponding to our empirical Fisher $\tilde{\mathbf{F}}_{\text{new}}(\boldsymbol{\mu})$ is different from the standard empirical Fisher considered by Kunstner et al. (2019). Indeed, some limitations in Kunstner et al. (2019) are due to the ill-conditioning issue mentioned in Sec. 2. Importantly, we estimate a preconditioner $\mathbf{S}$ and overcome the ill-conditioning issue by incorporating outer products from previous iterations (e.g., by a moving average) – even in the full-batch setting as shown in (5). Aggregating these previous outer products makes our approximation distinct from the approximation considered by Kunstner et al. (2019) where the authors do not make use of previous products. As shown in Fig. 10, our estimation works well in both convex examples constructed by Kunstner et al. (2019) and non-convex regression. Furthermore, our estimation has been theoretically justified (Hazan et al., 2006; Mukkamala & Hein, 2017; Wang et al., 2020) for root-free methods in (convex) settings similar to Kunstner et al. (2019). Thus, our empirical Fisher does not suffer from limitations of the standard empirical Fisher considered in Kunstner et al. (2019) when using our Fisher in the preconditioner update scheme.

3.3. Disentangling the Outer Product from the Fisher

At first glance, the outer product $\mathbf{H}_{\text{scaled}}$ coincides with another empirical Fisher $\tilde{\mathbf{F}}_{\text{incorrect}}(\boldsymbol{\mu}) = \nabla_\mu \log p_{\text{scaled}}(\mathbf{y} \mid \mathbf{X};\boldsymbol{\mu})\nabla_\mu^\top \log p_{\text{scaled}}(\mathbf{y} \mid \mathbf{X};\boldsymbol{\mu})$ in the scaled case. This Fisher is defined over a *different* joint distribution $p_{\text{scaled}}(\mathbf{y} \mid \mathbf{X};\boldsymbol{\mu}) = \prod_{i=1}^N p_{\text{scaled}}(y_i \mid \mathbf{x}_i;\boldsymbol{\mu})$ and depends on a *scaled* per-sample probability distribution over label y_i as $p_{\text{scaled}}(y_i \mid \mathbf{x}_i;\boldsymbol{\mu}) = \exp(-{}^1\!/_N c(y_i, f(\mathbf{x}_i;\boldsymbol{\mu})))$. However, this scaled distribution is not normalized. Therefore, we cannot define a valid FIM that corresponds to the outer product. In other words, the outer product is not an empirical Fisher matrix. The normalization condition of a per-sample distribution is often overlooked in the literature when considering a gradient outer product as an empirical Fisher.

4. Matrix Adaptive Methods

We develop a class of matrix adaptive methods without using matrix decompositions for training with mini-batches. This introduces another challenge: a dense matrix-valued preconditioner may be too costly to store. We address this by enforcing the preconditioner to be *structured*; specifically, Kronecker-factored. We use Eq. (5) to incorporate curvature information from previous iterations, especially for training with *mini-batches*. This is needed as we want to approximate full-batch curvature information. This curvature aggregation is also useful to handle the ill-conditioning issue mentioned in Sec. 2 when mini-batch curvature denoted by $\mathcal{H}$ is a gradient outer product $\mathbf{H}$. Unfortunately, Eq. (5) suggests that the optimizer's preconditioner $\mathbf{S}$ must have the same structure as the curvature approximation $\mathcal{H}$, so that the structure is maintained under the update. Thus, an additional projection step is needed when $\mathcal{H}$ has an incompatible structure. For instance, the outer product structure in $\mathcal{H}$ when $\mathcal{H} = \mathbf{H}$ is incompatible with the Kronecker-factored structure in $\mathbf{S}$. It is common practice to introduce a *sequential* inner loop to solve an optimization sub-problem at the projection step. However, this increases the iteration cost. Prime examples are the Frank-Wolfe method and Newton-CG (Martens & Sutskever, 2012). One idea to avoid using such an inner loop is to use a customized approximation such as Shampoo (Gupta et al., 2018) on a case-by-case

Shampoo

1: Compute $\mathbf{G} := \mathrm{Mat}(\nabla \ell_{\text{scaled}}(\boldsymbol{\mu}))$
$\hat{\mathbf{S}}_C \leftarrow (1-\beta_2\gamma)\hat{\mathbf{S}}_C + \beta_2 \mathbf{G}\mathbf{G}^\top$
$\hat{\mathbf{S}}_K \leftarrow (1-\beta_2\gamma)\hat{\mathbf{S}}_K + \beta_2 \mathbf{G}^\top\mathbf{G}$
2: $\mathrm{Mat}(\boldsymbol{\mu}) \leftarrow \mathrm{Mat}(\boldsymbol{\mu}) - \beta_1 \hat{\mathbf{S}}_C^{-1/4}\mathbf{G}\hat{\mathbf{S}}_K^{-1/4}$

Square-root-free Shampoo (Ours)

1: Compute $\mathbf{G} := \mathrm{Mat}(\nabla \ell_{\text{scaled}}(\boldsymbol{\mu}))$
$\mathbf{S}_C \leftarrow (1-\beta_2\gamma)\mathbf{S}_C + \frac{\beta_2}{d}(B\mathbf{G}\mathbf{S}_K^{-1}\mathbf{G}^\top)$
$\mathbf{S}_K \leftarrow (1-\beta_2\gamma)\mathbf{S}_K + \frac{\beta_2}{p}(B\mathbf{G}^\top\mathbf{S}_C^{-1}\mathbf{G})$
2: $\mathrm{Mat}(\boldsymbol{\mu}) \leftarrow \mathrm{Mat}(\boldsymbol{\mu}) - \beta_1 \mathbf{S}_C^{-1}\mathbf{G}\mathbf{S}_K^{-1}$

Inverse&root-free Shampoo (Ours)

1: Compute $\mathbf{G} := \mathrm{Mat}(\nabla \ell_{\text{scaled}}(\boldsymbol{\mu}))$
$\mathbf{C} \leftarrow \mathbf{C}\exp(-\frac{\beta_2}{2d}(B\mathbf{C}^\top\mathbf{G}\mathbf{K}\mathbf{K}^\top\mathbf{G}^\top\mathbf{C} - \gamma d\mathbf{I}))$
$\mathbf{K} \leftarrow \mathbf{K}\exp(-\frac{\beta_2}{2p}(B\mathbf{K}^\top\mathbf{G}^\top\mathbf{C}\mathbf{C}^\top\mathbf{G}\mathbf{K} - \gamma p\mathbf{I}))$
2: $\mathrm{Mat}(\boldsymbol{\mu}) \leftarrow \mathrm{Mat}(\boldsymbol{\mu}) - \beta_1 \mathbf{C}\mathbf{C}^\top\mathbf{G}\mathbf{K}\mathbf{K}^\top$

Figure 5. Structured matrix adaptive methods for a (scaled) loss function $\ell_{\text{scaled}}(\boldsymbol{\mu})$ defined by averaging over B data points in a mini-batch case. For simplicity, we assume $\mathrm{Mat}(\boldsymbol{\mu}) \in \mathbb{R}^{p\times d}$ is a weight matrix in a layer. $\hat{\mathbf{S}}_C, \mathbf{S}_C, \mathbf{C} \in \mathbb{R}^{p\times p}$ and $\hat{\mathbf{S}}_K, \mathbf{S}_K, \mathbf{K} \in \mathbb{R}^{d\times d}$ are non-singular matrices. In the inverse-free method, we directly update $\mathbf{C}$ and $\mathbf{K}$, and approximate the matrix exponential $\exp(\mathbf{N})$ by its first-order truncation $\exp(\mathbf{N}) \approx \mathbf{I} + \mathbf{N}$, where $\mathbf{C}$ and $\mathbf{S}_C$ are related as $\mathbf{S}_C^{-1} = \mathbf{C}\mathbf{C}^\top$ (c.f. Claim 10). $\mathbf{S}_K$ and $\mathbf{K}$ are similarly related. We initialize each of $\mathbf{C}$, $\mathbf{S}_C$, $\mathbf{S}_K$, and $\mathbf{K}$ to an identity matrix in our methods while each of $\hat{\mathbf{S}}_C$ and $\hat{\mathbf{S}}_K$ is initialized to zero (Gupta et al., 2018) . For simplicity, we do not include damping, weight decay, and momentum. See Fig. 7 in the Appendix for a full-fledged version.

basis. Instead, our *generic* approach aggregates *arbitrary* curvature information *without* an inner loop.

4.1. Square-root-free Methods through Gaussian Variational Approximations

We start with a dense preconditioner $\mathbf{S}$ to introduce the concept of preconditioner invariance which enables us to efficiently 'project' an arbitrary curvature approximation onto a flexible class of pre-conditioner parameterizations via the Bayesian learning rule (BLR, Khan et al., 2018; Lin et al., 2020; Khan & Rue, 2023) and the chain rule.

As discussed in Sec. 3, if the gradient outer product coincides with our empirical Fisher, we can view it as a Hessian approximation. This view allows us to extend the BLR originally developed for Newton's method. According to the BLR, we consider the preconditioner $\mathbf{S}$ in Equation (5) as an inverse covariance of a Gaussian, and the curvature approximation $\mathcal{H}$ as a partial derivative. By viewing the preconditioner as the inverse covariance, we gain new reparametrization invariance for the preconditioner $\mathbf{S}$ while preserving the affine invariance in $\boldsymbol{\mu}$ (c.f. Claim 2). This invariance allows us to reparameterize the preconditioner and obtain an equivalent update up to a first-order accuracy. One immediate application is to make matrix adaptive methods such as square-root-free full-AdaGrad *inverse-free* (see Eq. (18)) by reparameterizing the preconditioner, which is helpful for low-precision training. Thanks to the invariance, we can later impose sparse structures on the preconditioner (Lin et al., 2023a). This is flexible since preconditioner and curvature approximation can have independent structures.

Now, we describe the BLR. We consider a Bayesian problem formulation and solve a variational inference (VI) problem with a Gaussian variational approximation. In this setting, we consider NN weights as random variables and use a new symbol $\mathbf{w}$ to denote these weights as they are no longer learnable. We refer to $\boldsymbol{\mu}$ and $\mathbf{S}$ as the mean and the inverse covariance of the Gaussian $q(\mathbf{w} \mid \boldsymbol{\mu}, \mathbf{S})$ over weights $\mathbf{w}$. This VI problem (Barber & Bishop, 1997) (with $\gamma = 1$) is

$$\min_{\mu, S\succ 0} \mathcal{L}(\boldsymbol{\mu}, \mathbf{S}) := \mathbb{E}_{w\sim q}\big[\ell(\mathbf{w})\big] - \gamma \mathcal{Q}_q\,, \tag{11}$$

where $\ell(\mathbf{w})$ is the same loss defined in (1), $\mathbf{S} \succ 0$ must be positive-definite (PD), γ is the same hyperparameter as in (5), $q(\mathbf{w} \mid \boldsymbol{\mu}, \mathbf{S})$ is a Gaussian approximation with mean $\boldsymbol{\mu}$ and covariance $\mathbf{S}^{-1}$, and $\mathcal{Q}_q := \mathbb{E}_{w\sim q}[-\log q(\mathbf{w} \mid \boldsymbol{\mu}, \mathbf{S})] = -1/2 \log\det(\mathbf{S})$ is the Gaussian's entropy. When $\gamma = 0$, problem in Eq. (11) becomes a variational optimization (VO) problem (Staines & Barber, 2012; Khan et al., 2017) or a Gaussian adaptation problem[1] (Taxén & Kjellström, 1992; Bäck & Schwefel, 1993). Thus, the VI problem can be viewed as an entropy-regularized VO problem.

The FIM of the Gaussian $q(\mathbf{w} \mid \boldsymbol{\theta})$ under parameter $\boldsymbol{\theta}$ is

$$\mathbf{F}_{\text{gauss}}(\boldsymbol{\theta}) := -\mathbb{E}_{w\sim q}[\nabla^2_\theta \log q(\mathbf{w} \mid \boldsymbol{\theta})]\,, \tag{12}$$

has a closed-form expression and is PD whenever $\boldsymbol{\theta}$ is a valid parameterization. E.g., $\boldsymbol{\theta} := (\boldsymbol{\mu}, \mathbf{S})$ is a valid parameterization iff the inverse covariance $\mathbf{S}$ is PD. This FIM should not be confused with other Fisher matrices in Sec. 3.

We exploit a Riemannian geometric structure of the probabilistic (Gaussian) surrogate $q(\mathbf{w} \mid \boldsymbol{\theta})$ and, thus, use natural gradient descent (NGD) to solve the problem in Equation (11). An NGD step in parameterization $\boldsymbol{\theta}$ is

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta_1 (\mathbf{F}_{\text{gauss}}(\boldsymbol{\theta}))^{-1} \nabla_\theta \mathcal{L}\,. \tag{13}$$

Khan et al. (2018) show that this update is equivalent to

$$\begin{aligned} \mathbf{S} &\leftarrow \mathbf{S} + 2\beta_1 \partial_{S^{-1}}\mathcal{L} = (1-\beta_1\gamma)\mathbf{S} + \beta_1 \mathbb{E}_{w\sim q}[\nabla^2_w \ell(\mathbf{w})] \\ \boldsymbol{\mu} &\leftarrow \boldsymbol{\mu} - \beta_1 \mathbf{S}^{-1}\partial_\mu \mathcal{L} = \boldsymbol{\mu} - \beta_1 \mathbf{S}^{-1}\mathbb{E}_{w\sim q}[\nabla_w \ell(\mathbf{w})]\,, \end{aligned} \tag{14}$$

where we use the following identities[2] for the Gaussian $q(\mathbf{w} \mid \boldsymbol{\theta})$ (Opper & Archambeau, 2009; Lin et al., 2019b),

$$\partial_\mu \mathcal{L} = \mathbb{E}_{w\sim q}[\nabla_w \ell(\mathbf{w})],\ 2\partial_{S^{-1}}\mathcal{L} = \mathbb{E}_{w\sim q}[\nabla^2_w \ell(\mathbf{w})] - \gamma\mathbf{S}\,. \tag{15}$$

To recover the root-free update in Equation (5), we approximate the update in Equation (14) with

$$\mathbf{S} \leftarrow (1-\beta_2\gamma)\mathbf{S} + \beta_2 \overbrace{\mathcal{H}}^{\approx \nabla^2_\mu \ell(\boldsymbol{\mu})}\,,\ \boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \beta_1 \mathbf{S}^{-1} \overbrace{\mathbf{g}}^{=\nabla_\mu \ell(\boldsymbol{\mu})}\,, \tag{16}$$

[1] See Lin et al. (2021) for connections to gradient-free evolution methods (Wierstra et al., 2008; Glasmachers et al., 2010).

[2] See Lin et al. (2021) for sampling-based estimations to remove the need of computing $\nabla_w \ell(\mathbf{w})$ and $\nabla^2_w \ell(\mathbf{w})$. They are often used in gradient-free optimization and VI.

by [1] using a delta approximation at $\boldsymbol{\mu}$ to approximate the expectations highlighted in red, [2] replacing the Hessian with the outer product $\mathbf{H}$ as it is a Hessian approximation (e.g., $\mathcal{H} = \mathbf{H}$) in the unscaled case shown in Eq. (7), and [3] introducing an extra learning rate β_2 to compensate for the error of the curvature approximation. By using the delta approximation, we can use the NGD update derived for the Bayesian problem in Eq. (11) to solve the non-Bayesian problem in Eq. (1). This formulation[3] unveils the Gaussian approximation hidden in square-root-free updates as the NGD update recovers the square-root-free method. If the loss is scaled, we can replace the outer product with our empirical Fisher as a proper curvature approximation such as $\mathcal{H} = \tilde{\mathbf{F}}_{\text{scaled}}$ (c.f. Sec. 3). Moreover, we recover the update rule (Eq. (16)) to incorporate curvature information from previous iterations (e.g., by an exponentially moving average when $\gamma = 1$) for mini-batch training.

Examples Many square-root-free adaptive gradient methods can be derived from this update rule when the curvature approximation is the outer product (i.e., $\mathcal{H} = \mathbf{H}$). For example, Equation (16) becomes the update of square-root-free full-AdaGrad when $\gamma = 0$:

$$\mathbf{S} \leftarrow \mathbf{S} + \beta_2 \mathbf{H}, \;\; \boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \beta_1 \mathbf{S}^{-1} \mathbf{g} \,. \tag{17}$$

When $\mathbf{S} = \text{diag}(\mathbf{s})$ is diagonal, the update of (16) becomes

$$\mathbf{s} \leftarrow (1 - \beta_2 \gamma)\mathbf{s} + \beta_2 \text{diag}(\mathbf{H}), \;\; \boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \beta_1 \mathbf{S}^{-1} \mathbf{g} \,.$$

We obtain the root-free RMSProp update when $\gamma = 1$; likewise, the root-free diag-AdaGrad update when $\gamma = 0$.

Non-zero initialization The preconditioner $\mathbf{S}$ is often initialized to zero in adaptive methods. An immediate consequence of the BLR is that $\mathbf{S}$ should be initialized to a *non-zero* value because we view it as an inverse covariance. As shown in Fig. 2, this is important for the performance of root-free methods. Moreover, the updated $\mathbf{S}$ in Eq. (16) is guaranteed to be PD when $\mathbf{S}$ is initialized to a PD matrix since the product $\mathbf{H} = \mathbf{g}\mathbf{g}^\top$ is positive semi-definite. When $\mathcal{H}$ is an arbitrary approximation, we add a correction term to handle the PD constraint of $\mathbf{S}$ (Lin et al., 2020).

Inverse-free update via preconditioner invariance The BLR preserves the scale-invariance to the loss and affine reparametrization invariance in $\boldsymbol{\mu}$. The scale invariance comes from a proper Hessian approximation $\mathcal{H}$. The affine invariance in $\boldsymbol{\mu}$ is inherited from *invariance of natural-gradients* in the parameter space (e.g., $\boldsymbol{\theta} = (\boldsymbol{\mu}, \mathbf{S})$) of the Gaussian. Importantly, the BLR also introduces a new invariance to reparametrize preconditioner $\mathbf{S}$ thanks to the same invariance of natural-gradients of the Gaussian.

[3] See Lin et al. (2023b) for a manifold optimization formulation, where a Gaussian approximation with a non-constant mean is viewed as a PD submanifold. Newton's method can be viewed as Riemannian gradient descent on an augmented PD submanifold.

By exploiting this preconditioner invariance, we can easily make the root-free full-AdaGrad update inverse-free by performing NGD in another parameter space (e.g., $\boldsymbol{\eta} := (\boldsymbol{\mu}, \mathbf{S}^{-1})$). As shown in App. D, the update of $\mathbf{S}^{-1}$ is

$$\mathbf{S}^{-1} \leftarrow \mathbf{S}^{-1} - \beta_2 \mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1} + \tfrac{\beta_2^2}{2}\mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1}, \tag{18}$$

where we introduce a correction term highlighted in red to satisfy the PD constraint of $\mathbf{S}^{-1}$ (Lin et al., 2020). This update is inverse-free since it directly updates $\mathbf{S}^{-1}$. Moreover, it is scale and affine invariant, like the root-free AdaGrad update (c.f. Appx. F). We illustrate this preconditioner invariance by showing that the update for $\mathbf{S}^{-1}$, the root-free full-AdaGrad update for $\mathbf{S}$, and another inverse-free update for $\mathbf{A}$ (Lin et al., 2023b) are equivalent up to a first-order accuracy (c.f. Appx. E), where $\mathbf{S}^{-1} = \mathbf{A}\mathbf{A}^\top$. We can use this result to incorporate mini-batch curvature information (e.g., $\mathcal{H} = \mathbf{H}$) in a reparametrized space (e.g., $\mathbf{S}^{-1}$ or $\mathbf{A}$).

4.2. Decoupling Preconditioner & Curvature

We can further exploit the preconditioner invariance to allow preconditioner $\mathbf{S}$ and curvature $\mathcal{H}$ to have their own structures. From Eq. (14), we see that preconditioner $\mathbf{S}$ is the inverse covariance while the curvature approximation $\mathcal{H}$ appears as a term of partial derivative $2\partial_{S^{-1}}\mathcal{L} \approx \mathcal{H} - \gamma\mathbf{S}$ due to Eq. (15). If a sparse pattern in $\mathbf{S}$ can be obtained via reparameterization, we can perform NGD in this reparameterized space due to the parameterization invariance of NGs. This allows the curvature approximation $\mathcal{H}$ to have an independent structure and using the chain rule to 'project' the curvature approximation (as a partial derivative) onto the (sparse) reparameterized space of $\mathbf{S}$. Importantly, this neither introduces an inner loop nor significantly increases computational overhead. We will see an example next.

4.3. Kronecker-factored Adaptive Methods

Shampoo is a square-root-based Kronecker-factored method, where the inner (matrix) square roots are introduced due to a structural approximation and the outer (matrix) square root is inherited from full-AdaGrad. Shampoo's preconditioner $\mathbf{S}$ is coupled to its curvature approximation $\mathcal{H} \equiv \mathbf{H} = \mathbf{g}\mathbf{g}^\top$. Thus, it approximates the outer product $\mathbf{H} \approx \hat{\mathbf{S}}_C^{1/2} \otimes \hat{\mathbf{S}}_K^{1/2}$ and uses $\mathbf{S} = \left(\hat{\mathbf{S}}_C^{1/2} \otimes \hat{\mathbf{S}}_K^{1/2}\right)^{1/2}$ as preconditioner to ensure they have the same structure. Because an exponential moving average on $\mathbf{S}$ is infeasible due to the structural incompatibility between $\mathbf{H}$ and $\mathbf{S}$, Shampoo uses exponential moving averages over $\mathbf{S}_C$ and $\mathbf{S}_K$ as heuristics. Its update is

$$\hat{\mathbf{S}}_C \leftarrow (1 - \beta_2\gamma)\hat{\mathbf{S}}_C + \beta_2 \mathbf{G}\mathbf{G}^\top, \;\; \hat{\mathbf{S}}_K \leftarrow (1 - \beta_2\gamma)\hat{\mathbf{S}}_K + \beta_2 \mathbf{G}^\top\mathbf{G}$$
$$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \beta_1 \mathbf{S}^{-1}\mathbf{g} \iff \mathbf{M} \leftarrow \mathbf{M} - \beta_1 \hat{\mathbf{S}}_C^{-1/4} \mathbf{G} \hat{\mathbf{S}}_K^{-1/4},$$

where $\mathbf{G} := \text{Mat}(\mathbf{g})$ and $\mathbf{M} := \text{Mat}(\boldsymbol{\mu})$. This does not generalize to curvature approximations different from $\mathbf{g}\mathbf{g}^\top$.

In contrast, we consider a structural reparameterization of the inverse covariance $\mathbf{S} = \mathbf{S}_C \otimes \mathbf{S}_K$ as the preconditioner and perform NGD to update $\mathbf{S}_C$ and $\mathbf{S}_K$. We treat a curvature approximation $\mathcal{H}$, such as the gradient outer product $\mathbf{H}$,

as a partial derivative and use it to update $\mathbf{S}_C$ and $\mathbf{S}_K$ by the chain rule. We do not require the approximation $\mathcal{H}$ to have the same structure as the preconditioner $\mathbf{S}$. Thus, we decouple the preconditioner from the curvature approximation. Notably, our approach is root-free and does not require $\mathcal{H}$ to be an outer product—a key assumption of Shampoo. For example, we can use other inexpensive approximations $\mathcal{H}$ such as KFAC (Martens & Grosse, 2015).

Concretely, consider a parameterization $\boldsymbol{\tau} := (\boldsymbol{\mu}, \mathbf{S}_C, \mathbf{S}_K)$ of the Gaussian, where $\mathbf{S}_C \in \mathbb{R}^{p\times p}$ and $\mathbf{S}_K \in \mathbb{R}^{d\times d}$ are PD. This Gaussian is known as a matrix Gaussian[4]. The exact FIM $\mathbf{F}_{\text{gauss}}(\boldsymbol{\tau}) = \begin{bmatrix} \mathbf{F}_{\mu\mu} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_{CC} & \mathbf{F}_{CK} \\ \mathbf{0} & \mathbf{F}_{KC} & \mathbf{F}_{KK} \end{bmatrix}$ under this parameterization is singular since the Kronecker product is not unique. We consider a block-diagonal approximated FIM of the Gaussian denoted by $\tilde{\mathbf{F}}_{\text{gauss}}(\boldsymbol{\tau}) = \begin{bmatrix} \mathbf{F}_{\mu\mu} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_{CC} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{F}_{KK} \end{bmatrix}$ so that we can use other approximations $\mathcal{H}$ beyond the outer product. This approximate FIM is non-singular and well-defined by ignoring cross-terms $\mathbf{F}_{CK}$ and $\mathbf{F}_{KC}$ in the original FIM. We propose to perform approximate NGD with $\tilde{\mathbf{F}}_{\text{gauss}}(\boldsymbol{\tau})$

$$\begin{bmatrix} \boldsymbol{\mu} \\ \mathbf{S}_C \\ \mathbf{S}_K \end{bmatrix} \leftarrow \begin{bmatrix} \boldsymbol{\mu} \\ \mathbf{S}_C \\ \mathbf{S}_K \end{bmatrix} - \beta_1 \begin{bmatrix} \mathbf{S} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\frac{d}{2}\frac{\partial \mathbf{S}_C^{-1}}{\partial \mathbf{S}_C} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\frac{p}{2}\frac{\partial \mathbf{S}_K^{-1}}{\partial \mathbf{S}_K} \end{bmatrix}^{-1} \begin{bmatrix} \partial_\mu \mathcal{L} \\ \partial_{S_C} \mathcal{L} \\ \partial_{S_K} \mathcal{L} \end{bmatrix}$$
$$= \begin{bmatrix} \boldsymbol{\mu} - \beta_1 \mathbf{S}^{-1} \partial_\mu \mathcal{L} \\ \mathbf{S}_C + \frac{2\beta_1}{d} \partial_{S_C^{-1}} \mathcal{L} \\ \mathbf{S}_K + \frac{2\beta_1}{p} \partial_{S_K^{-1}} \mathcal{L} \end{bmatrix},$$

where $\mathbf{F}_{\mu\mu} = \mathbf{S} = \mathbf{S}_C \otimes \mathbf{S}_K$, $\mathbf{F}_{CC} = -\frac{d}{2}\partial \mathbf{S}_C^{-1}/\partial \mathbf{S}_C$, $\mathbf{F}_{KK} = -\frac{p}{2}\partial \mathbf{S}_K^{-1}/\partial \mathbf{S}_K$ are computed according to Eq.(12) or block-wise Bregman duality (Lin et al., 2019a). Then, we can use the chain rule to project the curvature $\mathcal{H}$ as a term of derivatives highlighted in blue

$$2\partial_{S_C^{-1}} \mathcal{L} = 2\frac{\partial \mathbf{S}^{-1}}{\partial \mathbf{S}_C^{-1}} \partial_{S^{-1}} \mathcal{L} \approx \frac{\partial \mathbf{S}^{-1}}{\partial \mathbf{S}_C^{-1}} [\mathcal{H} - \gamma \mathbf{S}_C \otimes \mathbf{S}_K]$$
$$2\partial_{S_K^{-1}} \mathcal{L} = 2\frac{\partial \mathbf{S}^{-1}}{\partial \mathbf{S}_K^{-1}} \partial_{S^{-1}} \mathcal{L} \approx \frac{\partial \mathbf{S}^{-1}}{\partial \mathbf{S}_K^{-1}} [\mathcal{H} - \gamma \mathbf{S}_C \otimes \mathbf{S}_K],$$

and update a Kronecker-factored preconditioner $\mathbf{S}$ without square-roots, where $2\partial_{S^{-1}} \mathcal{L} \approx \mathcal{H} - \gamma \mathbf{S} = \mathcal{H} - \gamma \mathbf{S}_C \otimes \mathbf{S}_K$ is due to Eq. 15. When $\mathcal{H} = \mathbf{H}$, we obtain a root-free Shampoo-like update rule due to the simplification of the derivatives:

$$2\partial_{S_C^{-1}} \mathcal{L} \approx \frac{\partial \mathbf{S}^{-1}}{\partial \mathbf{S}_C^{-1}} [\mathbf{H} - \gamma \mathbf{S}_C \otimes \mathbf{S}_K] = \mathbf{G}\mathbf{S}_K^{-1}\mathbf{G}^\top - \gamma d \mathbf{S}_C$$
$$2\partial_{S_K^{-1}} \mathcal{L} \approx \frac{\partial \mathbf{S}^{-1}}{\partial \mathbf{S}_K^{-1}} [\mathbf{H} - \gamma \mathbf{S}_C \otimes \mathbf{S}_K] = \mathbf{G}^\top \mathbf{S}_C^{-1}\mathbf{G} - \gamma p \mathbf{S}_K,$$

Notably, our root-free updates for $\mathbf{S}_K$ and $\mathbf{S}_C$ (c.f. Fig. 5) are correlated while Shampoo's updates for $\hat{\mathbf{S}}_K$ and $\hat{\mathbf{S}}_C$ are uncorrelated due to Shampoo's block-diagonal approximation of $\mathbf{H}$. When the loss is scaled, we obtain a similar update using our empirical Fisher (c.f. Sec. 3).

[4]We can also generalize our update to n-dimensional tensors using a tensor Gaussian (Ohlson et al., 2013).

This update is similar to other root-free second-order methods based on a multi-linear exponential family (Lin et al., 2019a) or a tensor Gaussian distribution (Ren & Goldfarb, 2021) when using the outer product as a curvature approximation. Unlike these works, our update is both affine-invariant and scale-invariant of the loss, and is applicable for other curvature approximations $\mathcal{H}$.

Structured inverse-free (IF) update To enable our update in half-precision, we consider an IF update by reparameterizing $\mathbf{S}^{-1} = \mathbf{S}_C^{-1} \otimes \mathbf{S}_K^{-1} = (\mathbf{C}\mathbf{C}^\top) \otimes (\mathbf{K}\mathbf{K}^\top)$ and updating $\mathbf{C}$ and $\mathbf{K}$ instead of $\mathbf{S}_C$ and $\mathbf{S}_K$ (Lin et al., 2023b) and exploit the preconditioner invariance. This update is IF and root-free (c.f. Fig. 5), avoiding numerically unstable matrix inversions and decompositions (c.f. Appx. G for a derivation). Moreover, this update is equivalent to updating $\mathbf{S}_C$ and $\mathbf{S}_K$ up to a first-order accuracy (c.f. Appx. I). Last but not least, this update also allows the use of sparse factors $\mathbf{C}$ and $\mathbf{K}$ (Lin et al., 2023a) to further lower memory consumption. We can construct an expressive structured preconditioner $\mathbf{S}_C$ or its inverse $\mathbf{S}_C^{-1}$ using a sparse $\mathbf{C}$.

We find that IF-Shampoo performs similarly to Shampoo regarding per-iteration progress on modern vision models such as vision transformers and mambas (see Fig. 8). However, our method can run in BFP-16, in contrast to Shampoo. We observe that one step of IF-Shampoo takes less than half the time of Shampoo (see Fig. 3) and requires less memory. This promising result will make matrix adaptive methods more prominent in modern large-scale training.

5. Conclusion

We investigated root-free adaptive methods and strengthened their motivation from a second-order perspective. Surprisingly, we found empirically that removing the root not only *closes* the generalization gap between adaptive methods and SGD on CNNs, but also *maintains* the performance of root-based methods on vision attention models. Removing the root eliminates the connection to sign descent which has been hypothesized to cause the gap on convolutional NNs and transformers. However, our findings highlight that adaptivity might be an important concept for the success of such methods that is currently overlooked, which poses new questions regarding the role of adaptivity and the understanding of adaptive methods. Conceptually, we established a second-order view on root-free methods by viewing their gradient outer product as a novel empirical Fisher that preserves scale-invariance of the loss and affine invariance. Computationally, we introduced the concept of preconditioner invariance to reparametrize preconditioners and developed matrix adaptive, inverse-free methods that stably work in BFP-16 and train faster than their root-based counterpart. By using low-precision training, our IF-Shampoo has a similar iteration cost as diagonal methods while achieving better performance on modern NN models.

Acknowledgements

We thank Emtiyaz Khan, Mark Schmidt, Kirill Neklyudov, and Roger Grosse for helpful discussions at the early stage of this work. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute. Runa Eschenhagen is supported by ARM and the Cambridge Trust. Richard E. Turner is supported by Google, Amazon, ARM, Improbable and EPSRC grant EP/T005386/1.

Impact Statement

This paper introduces research aimed at improving deep learning optimization. While there are numerous potential societal implications stemming from our work, such as giving the ability to train neural networks more efficiently, we believe that none warrant specific emphasis within this context.

References

Agarwal, N., Bullins, B., Chen, X., Hazan, E., Singh, K., Zhang, C., and Zhang, Y. Efficient full-matrix adaptive regularization. In *International Conference on Machine Learning*, pp. 102–110. PMLR, 2019.

Anil, R., Gupta, V., Koren, T., Regan, K., and Singer, Y. Scalable second order optimization for deep learning. *arXiv preprint arXiv:2002.09018*, 2020.

Bäck, T. and Schwefel, H.-P. An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, 1(1):1–23, 1993.

Balles, L. and Hennig, P. Dissecting adam: The sign, magnitude and variance of stochastic gradients. In *International Conference on Machine Learning*, pp. 404–413. PMLR, 2018.

Barber, D. and Bishop, C. Ensemble learning for multi-layer networks. *Advances in neural information processing systems*, 10, 1997.

Becker, S., Le Cun, Y., et al. Improving the convergence of back-propagation learning with second order methods. In *Proceedings of the 1988 connectionist models summer school*, pp. 29–37, 1988.

Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

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

Chen, X., Liang, C., Huang, D., Real, E., Wang, K., Liu, Y., Pham, H., Dong, X., Luong, T., Hsieh, C.-J., et al. Symbolic discovery of optimization algorithms. *arXiv preprint arXiv:2302.06675*, 2023.

Choi, D., Shallue, C. J., Nado, Z., Lee, J., Maddison, C. J., and Dahl, G. E. On empirical comparisons of optimizers for deep learning. *arXiv preprint arXiv:1910.05446*, 2019.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Glasmachers, T., Schaul, T., Yi, S., Wierstra, D., and Schmidhuber, J. Exponential natural evolution strategies. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pp. 393–400, 2010.

Gupta, V., Koren, T., and Singer, Y. Shampoo: Preconditioned Stochastic Tensor Optimization. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1842–1850, 2018.

Hatamizadeh, A., Yin, H., Heinrich, G., Kautz, J., and Molchanov, P. Global context vision transformers. In *International Conference on Machine Learning*, pp. 12633–12646. PMLR, 2023.

Hazan, E., Agarwal, A., and Kale, S. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69:169–192, 2006.

Hui, L. and Belkin, M. Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks. *arXiv preprint arXiv:2006.07322*, 2020.

Khan, M. E. and Rue, H. The bayesian learning rule. *Journal of Machine Learning Research*, 24(281):1–46, 2023.

Khan, M. E., Lin, W., Tangkaratt, V., Liu, Z., and Nielsen, D. Variational adaptive-Newton method for explorative learning. *arXiv preprint arXiv:1711.05560*, 2017.

Khan, M. E., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y., and Srivastava, A. Fast and scalable Bayesian deep learning by weight-perturbation in Adam. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 2611–2620, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kunstner, F., Hennig, P., and Balles, L. Limitations of the empirical fisher approximation for natural gradient descent. *Advances in neural information processing systems*, 32, 2019.

Kunstner, F., Chen, J., Lavington, J. W., and Schmidt, M. Noise is not the main factor behind the gap between sgd and adam on transformers, but sign descent might be. *arXiv preprint arXiv:2304.13960*, 2023.

Lin, W., Khan, M. E., and Schmidt, M. Fast and simple natural-gradient variational inference with mixture of exponential-family approximations. In *International Conference on Machine Learning*, pp. 3992–4002, 2019a.

Lin, W., Khan, M. E., and Schmidt, M. Stein's Lemma for the Reparameterization Trick with Exponential-family Mixtures. *arXiv preprint arXiv:1910.13398*, 2019b.

Lin, W., Schmidt, M., and Khan, M. E. Handling the positive-definite constraint in the bayesian learning rule. In *International Conference on Machine Learning*, pp. 6116–6126. PMLR, 2020.

Lin, W., Nielsen, F., Emtiyaz, K. M., and Schmidt, M. Tractable structured natural-gradient descent using local parameterizations. In *International Conference on Machine Learning*, pp. 6680–6691. PMLR, 2021.

Lin, W., Dangel, F., Eschenhagen, R., Neklyudov, K., Kristiadi, A., Turner, R. E., and Makhzani, A. Structured inverse-free natural gradient: Memory-efficient & numerically-stable kfac for large neural nets. *arXiv preprint arXiv:2312.05705*, 2023a.

Lin, W., Duruisseaux, V., Leok, M., Nielsen, F., Khan, M. E., and Schmidt, M. Simplifying momentum-based positive-definite submanifold optimization with applications to deep learning. In *International Conference on Machine Learning*, pp. 21026–21050. PMLR, 2023b.

Liu, H., Li, Z., Hall, D., Liang, P., and Ma, T. Sophia: A scalable stochastic second-order optimizer for language model pre-training. *arXiv preprint arXiv:2305.14342*, 2023.

Liu, Y., Tian, Y., Zhao, Y., Yu, H., Xie, L., Wang, Y., Ye, Q., and Liu, Y. Vmamba: Visual state space model. *arXiv preprint arXiv:2401.10166*, 2024.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Martens, J. New insights and perspectives on the natural gradient method. *The Journal of Machine Learning Research*, 21(1):5776–5851, 2020.

Martens, J. and Grosse, R. Optimizing neural networks with Kronecker-factored approximate curvature. In *International Conference on Machine Learning*, pp. 2408–2417, 2015.

Martens, J. and Sutskever, I. Training deep and recurrent networks with hessian-free optimization. In *Neural Networks: Tricks of the Trade: Second Edition*, pp. 479–535. Springer, 2012.

Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

Mukkamala, M. C. and Hein, M. Variants of rmsprop and adagrad with logarithmic regret bounds. 2017.

Nesterov, Y. and Nemirovskii, A. *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.

Ohlson, M., Ahmad, M. R., and Von Rosen, D. The multilinear normal distribution: Introduction and some basic properties. *Journal of Multivariate Analysis*, 113:37–47, 2013.

Opper, M. and Archambeau, C. The variational Gaussian approximation revisited. *Neural computation*, 21(3):786–792, 2009.

Pascanu, R. and Bengio, Y. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

Reddi, S. J., Kale, S., and Kumar, S. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

Ren, Y. and Goldfarb, D. Tensor normal training for deep learning models. *Advances in Neural Information Processing Systems*, 34:26040–26052, 2021.

Roux, N., Manzagol, P.-A., and Bengio, Y. Topmoumoute online natural gradient algorithm. *Advances in neural information processing systems*, 20, 2007.

Shi, H.-J. M., Lee, T.-H., Iwasaki, S., Gallego-Posada, J., Li, Z., Rangadurai, K., Mudigere, D., and Rabbat, M. A distributed data-parallel pytorch implementation of the distributed shampoo optimizer for training neural networks at-scale. *arXiv preprint arXiv:2309.06497*, 2023.

Staines, J. and Barber, D. Variational optimization. *arXiv preprint arXiv:1212.4507*, 2012.

Taxén, L. and Kjellström, G. Gaussian adaptation, an evolution-based efficient global optimizer. 1992.

Tieleman, T. and Hinton, G. Rmsprop: Divide the gradient by a running average of its recent magnitude. *Coursera*, 2012.

Wang, G., Lu, S., Cheng, Q., Tu, W., and Zhang, L. Sadam: A variant of adam for strongly convex functions. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rye5YaEtPr.

Wierstra, D., Schaul, T., Peters, J., and Schmidhuber, J. Natural evolution strategies. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 3381–3387. IEEE, 2008.

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., and Recht, B. The marginal value of adaptive gradient methods in machine learning. *Advances in neural information processing systems*, 30, 2017.

Yang, J., Li, C., Dai, X., and Gao, J. Focal modulation networks. *Advances in Neural Information Processing Systems*, 35:4203–4217, 2022.

Zhang, J., Karimireddy, S. P., Veit, A., Kim, S., Reddi, S., Kumar, S., and Sra, S. Why are adaptive methods good for attention models? *Advances in Neural Information Processing Systems*, 33:15383–15393, 2020.

Zhang, W., Yin, Z., Sheng, Z., Li, Y., Ouyang, W., Li, X., Tao, Y., Yang, Z., and Cui, B. Graph attention multi-layer perceptron. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4560–4570, 2022.

Square-root-free RMSProp

Compute gradient $\mathbf{g} \coloneqq \nabla \ell_{\text{scaled}}(\boldsymbol{\mu})$
$\mathbf{s} \leftarrow (1-\beta_2)\mathbf{s} + \beta_2 B \mathbf{g}^2$
$\mathbf{m} \leftarrow \alpha_1 \mathbf{m} + \mathbf{g}/(\mathbf{s}+\lambda) + \kappa \boldsymbol{\mu}$
$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \beta_1 \mathbf{m}$

Figure 6. A full-fledged version of our square-root-free RMSProp, where α_1, λ, κ are the weight to include momentum, damping, weight decay, respectively. The preconditioner $\mathbf{s}$ is initialized with 1.

Square-root-free Shampoo (Ours)

Compute $\mathbf{G} \coloneqq \text{Mat}(\nabla \ell_{\text{scaled}}(\boldsymbol{\mu}))$
$\mathbf{S}_C \leftarrow (1-\beta_2\gamma)\mathbf{S}_C + \frac{\beta_2}{d}(B\mathbf{G}\mathbf{S}_K^{-1}\mathbf{G}^\top + \lambda \text{Tr}(\mathbf{S}_K^{-1})\mathbf{I}_p)$
$\mathbf{S}_K \leftarrow (1-\beta_2\gamma)\mathbf{S}_K + \frac{\beta_2}{p}(B\mathbf{G}^\top\mathbf{S}_C^{-1}\mathbf{G} + \lambda \text{Tr}(\mathbf{S}_C^{-1})\mathbf{I}_d)$
$\mathbf{M} \leftarrow \alpha_1 \mathbf{M} + \mathbf{S}_C^{-1}\mathbf{G}\mathbf{S}_K^{-1} + \kappa \text{Mat}(\boldsymbol{\mu})$
$\text{Mat}(\boldsymbol{\mu}) \leftarrow \text{Mat}(\boldsymbol{\mu}) - \beta_1 \mathbf{M}$

Inverse-free Shampoo (Ours)

Compute $\mathbf{G} \coloneqq \text{Mat}(\nabla \ell_{\text{scaled}}(\boldsymbol{\mu}))$
$\mathbf{m}_C \leftarrow \alpha_2 \mathbf{m}_C + \frac{(1-\alpha_2)}{2d}(B\mathbf{C}^\top\mathbf{G}\mathbf{K}\mathbf{K}^\top\mathbf{G}^\top\mathbf{C} + \lambda \text{Tr}(\mathbf{K}\mathbf{K}^\top)\mathbf{C}^\top\mathbf{C} - d\gamma\mathbf{I}_p)$
$\mathbf{m}_K \leftarrow \alpha_2 \mathbf{m}_K + \frac{(1-\alpha_2)}{2p}(B\mathbf{K}^\top\mathbf{G}^\top\mathbf{C}\mathbf{C}^\top\mathbf{G}\mathbf{K} + \lambda \text{Tr}(\mathbf{C}\mathbf{C}^\top)\mathbf{K}^\top\mathbf{K} - p\gamma\mathbf{I}_d)$
$\mathbf{C} \leftarrow \mathbf{C}\exp(-\beta_2 \mathbf{m}_C)$
$\mathbf{K} \leftarrow \mathbf{K}\exp(-\beta_2 \mathbf{m}_K)$
$\mathbf{M} \leftarrow \alpha_1 \mathbf{M} + \mathbf{C}\mathbf{C}^\top\mathbf{G}\mathbf{K}\mathbf{K}^T + \kappa \text{Mat}(\boldsymbol{\mu})$
$\text{Mat}(\boldsymbol{\mu}) \leftarrow \text{Mat}(\boldsymbol{\mu}) - \beta_1 \mathbf{M}$

Figure 7. A full-fledged version of our matrix methods, where α_1, λ, κ are the weight to include momentum, damping, weight decay, respectively. We also include weight α_2 for Riemannian momentum suggested by Lin et al. (2023b). We approximate the matrix exponential $\exp(\mathbf{N})$ by its first-order truncation $\exp(\mathbf{N}) \approx \mathbf{I} + \mathbf{N}$ as suggested by Lin et al. (2023b). For our inverse-free Shampoo, we store and update preconditioner factors $\mathbf{C}$, $\mathbf{m}_C$, $\mathbf{K}$, $\mathbf{m}_K$ in half precision. For numerical stability in half precision, we update $\mathbf{C}$ and $\mathbf{K}$ as $\mathbf{C} \leftarrow \mathbf{C} - \frac{\beta \mathbf{m}_C}{\max\{\|\mathbf{m}_C\|,1\}}$ and $\mathbf{K} \leftarrow \mathbf{K} - \frac{\beta \mathbf{m}_K}{\max\{\|\mathbf{m}_K\|,1\}}$, where $\|\cdot\|$ denotes the Frobenius norm. Using the matrix norm also allows us to make discretization errors (e.g., the term $O(\beta_2^2)$ in Claim 10) in the updates negligible, where $\beta_2 \coloneqq \frac{\beta}{\max\{\|\mathbf{m}_C\|,1\}}$ and $\alpha_2 = 0$ is the case considered in the Claim.

A. Example: Affine Invariance of Root-Free Methods

We demonstrate the affine invariance by an example and show how adding the root breaks the invariance. Consider a loss function $l_a(a) = \frac{1}{2}a^2$ with an initial point $a_0 = 2$, a root-based update is $a_{\text{new}} = a_0 - s_a^{-1} g_a = 2 - 1 = 1$, where gradient $g_a = \nabla_a l_a(a) = a_0 = 2$ and preconditioner $s_a = \sqrt{g_a^2} = |a_0| = 2$. Now, consider a reparameterized loss as $l_2(b) = \frac{1}{2}(2b)^2$ with an initial point b_0, where $a = 2b$. Thus, $b_0 = 1$ when $a_0 = 2$. The update becomes $b_{\text{new}} = b_0 - s_b^{-1} g_b = 1 - 1 = 0$, where gradient $g_b = \nabla_b l_b(b) = 4b_0 = 4$ and preconditioner $s_b = \sqrt{g_b^2} = 4|b_0| = 4$. Unfortunately, the updated $a_{\text{new}} = 1$ is not equivalent to the updated $b_{\text{new}} = 0$ since $a_{\text{new}} \neq 2b_{\text{new}}$.

Now, consider a square-root-free update for the original loss as $a_{\text{new}} = a_0 - s_a^{-1} g_a = 2 - 0.5 = 1.5$, where gradient $g_a = \nabla_a l_a(a) = a_0 = 2$ and preconditioner $s_a = g_a^2 = a_0^2 = 4$. Similarly, the update for the reparameterized loss is $b_{\text{new}} = b_0 - s_b^{-1} g_b = 1 - 0.25 = 0.75$, where gradient $g_b = \nabla_b l_b(b) = 4b_0 = 4$ and preconditioner $s_b = g_b^2 = 16b_0^2 = 16$. Note that the updated $a_{\text{new}} = 1.5$ is equivalent to the updated $b_{\text{new}} = 0.75$ since $a_{\text{new}} = 2b_{\text{new}}$. Thus, removing the root preserves the affine invariance.

B. Proof of Claim 1

We first show that our Fisher matrix coincides with the standard Fisher.

$$\mathbf{F}_{\text{new}}(\boldsymbol{\mu}) = \mathbf{F}_{\text{standard}}(\boldsymbol{\mu}) = \sum_{i=1}^{N} E_{y_i \sim p}\left[\nabla_\mu \log p(y_i|\mathbf{x}_i;\boldsymbol{\mu}) \nabla_\mu^\top \log p(y_i|\mathbf{x}_i;\boldsymbol{\mu})\right] \tag{19}$$

Similarly, we can show our mini-batch Fisher coincides with the standard mini-batch Fisher.

$$\mathbf{F}_{\text{mini}}(\boldsymbol{\mu}) = \mathbf{F}_{\text{standard-mini}}(\boldsymbol{\mu}) \coloneqq \sum_{i=1}^{B} E_{y_i\sim p}\big[\nabla_\mu \log p(y_i|\mathbf{x}_i;\boldsymbol{\mu})\nabla_\mu^\top \log p(y_i|\mathbf{x}_i;\boldsymbol{\mu})\big] \tag{20}$$

Thus, it is easy to see that $\frac{1}{B}\mathbf{F}_{\text{mini}}(\boldsymbol{\mu})$ is an unbiased estimation of $\frac{1}{N}\mathbf{F}_{\text{new}}(\boldsymbol{\mu})$ since the standard mini-batch Fisher is an unbiased estimation of the standard full-batch Fisher.

Now, we show that our Fisher matrix coincides with the standard Fisher. Recall that we define the joint distribution over labels is $p(\mathbf{y}|\mathbf{X};\boldsymbol{\mu}) = \prod_{i=1}^N p(y_i|\mathbf{x}_i;\boldsymbol{\mu})$

$$\begin{aligned}
&\mathbf{F}_{\text{new}}(\boldsymbol{\mu})\\
=&E_{\mathbf{y}\sim p}\big[\nabla_\mu \log p(\mathbf{y}|\mathbf{X};\boldsymbol{\mu})\nabla_\mu^\top \log p(\mathbf{y}|\mathbf{X};\boldsymbol{\mu})\big]\\
=&E_{\mathbf{y}\sim p}\big[\sum_i \big(\nabla_\mu \log p(y_i|\mathbf{x}_i;\boldsymbol{\mu})\big)\big(\sum_j \nabla_\mu^\top \log p(y_j|\mathbf{x}_j;\boldsymbol{\mu})\big)\big]\\
=&E_{\mathbf{y}\sim p}\big[\sum_{i=j} \big(\nabla_\mu \log p(y_i|\mathbf{x}_i;\boldsymbol{\mu})\big)\big(\nabla_\mu^\top \log p(y_j|\mathbf{x}_j;\boldsymbol{\mu})\big)\big] + E_{\mathbf{y}\sim p}\big[\sum_{i\neq j} \big(\nabla_\mu \log p(y_i|\mathbf{x}_i;\boldsymbol{\mu})\big)\big(\nabla_\mu^\top \log p(y_j|\mathbf{x}_j;\boldsymbol{\mu})\big)\big]\\
=&\sum_i E_{y_i\sim p}\big[\big(\nabla_\mu \log p(y_i|\mathbf{x}_i;\boldsymbol{\mu})\big)\big(\nabla_\mu^\top \log p(y_i|\mathbf{x}_i;\boldsymbol{\mu})\big)\big] + E_{\mathbf{y}\sim p}\big[\sum_{i\neq j} \big(\nabla_\mu \log p(y_i|\mathbf{x}_i;\boldsymbol{\mu})\big)\big(\nabla_\mu^\top \log p(y_j|\mathbf{x}_j;\boldsymbol{\mu})\big)\big]\\
=&\mathbf{F}_{\text{standard}}(\boldsymbol{\mu}) + \sum_{i\neq j} E_{\mathbf{y}\sim p}\big[\big(\nabla_\mu \log p(y_i|\mathbf{x}_i;\boldsymbol{\mu})\big)\big(\nabla_\mu^\top \log p(y_j|\mathbf{x}_j;\boldsymbol{\mu})\big)\big]\\
=&\mathbf{F}_{\text{standard}}(\boldsymbol{\mu})
\end{aligned}$$

where the last line is due to the independence and normalization of per-sample distributions as shown below.

Since each per-sample distribution is independent, we have

$$\sum_{i\neq j} E_{\mathbf{y}\sim p}\big[\big(\nabla_\mu \log p(y_i|\mathbf{x}_i;\boldsymbol{\mu})\big)\big(\nabla_\mu^\top \log p(y_j|\mathbf{x}_j;\boldsymbol{\mu})\big)\big] = \sum_{i\neq j} \underbrace{E_{y_i\sim p}\big[\nabla_\mu \log p(y_i|\mathbf{x}_i;\boldsymbol{\mu})\big]}_{=\mathbf{0}} E_{y_j\sim p}\big[\nabla_\mu^\top \log p(y_j|\mathbf{x}_j;\boldsymbol{\mu})\big] = \mathbf{0}$$

where we make use of the following result as the per-sample distribution is normalized.

$$E_{y_i\sim p}\big[\nabla_\mu \log p(y_i|\mathbf{x}_i;\boldsymbol{\mu})\big] = \int \nabla_\mu p(y_i|\mathbf{x}_i;\boldsymbol{\mu})dy_i = \nabla_\mu \underbrace{\int p(y_i|\mathbf{x}_i;\boldsymbol{\mu})dy_i}_{=1} = \mathbf{0}.$$

C. Proof of Claim 2

Recall that the (unscaled) optimization problem in (1) is

$$\min_\mu \ell(\boldsymbol{\mu}) \tag{21}$$

Now, consider reparametrizing $\boldsymbol{\mu}$ with a known non-singular matrix $\mathbf{A}$ and a constant vector $\mathbf{c}$ as $\boldsymbol{\mu} = \mathbf{A}\mathbf{m} + \mathbf{c}$. In this case, the optimization problem becomes

$$\min_m \ell^{\text{rep}}(\mathbf{m}) \coloneqq \ell(\mathbf{A}\mathbf{m} + \mathbf{c}) \tag{22}$$

We will show that a square-root-free method is affine invariant at each step. In other words, if we use the same square-root-free method to solve these two problems, they are equivalent.

For the first problem, the method takes the following step at iteration t

$$\mathbf{S}_{t+1} = (1-\beta_2\gamma)\mathbf{S}_t + \beta_2\mathbf{H}_t, \quad \boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \beta_1\mathbf{S}_{t+1}^{-1}\mathbf{g}_t \tag{23}$$

where $\mathbf{g}_t \coloneqq \nabla_\mu \ell(\boldsymbol{\mu})\big|_{\mu=\mu_t}$ and $\mathbf{H}_t = \mathbf{g}_t\mathbf{g}_t^\top$.

For the second problem, we assume $\mathbf{S}_0^{\text{rep}}$ is initialized with $\mathbf{A}^\top \mathbf{S}_0 \mathbf{A}$ and $\mathbf{A}^{-1}(\boldsymbol{\mu}_0 - \mathbf{c}) = \mathbf{m}_0$ since $\mathbf{A}$ and $\mathbf{c}$ are known. In this case, the square-root-free update at the first iteration becomes

$$\mathbf{S}_1^{\text{rep}} = (1-\beta_2\gamma)\mathbf{S}_0^{\text{rep}} + \beta_2 \mathbf{H}_0^{\text{rep}} = (1-\beta_2\gamma)\mathbf{S}_0^{\text{rep}} + \beta_2 \mathbf{A}^\top \mathbf{H}_0 \mathbf{A} = \mathbf{A}^\top \underbrace{\big((1-\beta_2\gamma)\mathbf{S}_0 + \beta_2 \mathbf{H}_0\big)}_{=\mathbf{S}_1} \mathbf{A}$$

$$\begin{aligned}\mathbf{m}_1 &= \mathbf{m}_0 - \beta_1 \big(\mathbf{S}_1^{\text{rep}}\big)^{-1} \mathbf{g}_0^{\text{rep}} = \mathbf{m}_0 - \beta_1 \mathbf{A}^{-1}\mathbf{S}_1^{-1}\mathbf{A}^{-T}\mathbf{A}^\top \mathbf{g}_0 = \mathbf{A}^{-1}(\boldsymbol{\mu}_0 - \mathbf{c}) - \beta_1 \mathbf{A}^{-1}\mathbf{S}_1^{-1}\mathbf{g}_0 \\ &= \mathbf{A}^{-1}(\boldsymbol{\mu}_0 - \mathbf{c} - \beta_1 \mathbf{S}_1^{-1}\mathbf{g}_0) = \mathbf{A}^{-1}(\boldsymbol{\mu}_1 - \mathbf{c})\end{aligned}$$

where we use the following identities when $t = 0$.

$$\begin{aligned}\mathbf{g}_t^{\text{rep}} &\coloneqq \nabla_m \ell^{\text{rep}}(\mathbf{m})\big|_{m=m_t} = \frac{\partial \boldsymbol{\mu}}{\partial \mathbf{m}}|_{m=m_t} \nabla_\mu \ell^{\text{rep}}(\mathbf{m})\big|_{\mu=\mu_t} = \mathbf{A}^\top \nabla_\mu \ell(\boldsymbol{\mu})|_{\mu=\mu_t} = \mathbf{A}^\top \mathbf{g}_t \\ \mathbf{H}_t^{\text{rep}} &\coloneqq \mathbf{g}_t^{\text{rep}} \big(\mathbf{g}_t^{\text{rep}}\big)^\top = \mathbf{A}^\top \mathbf{H}_t \mathbf{A}\end{aligned}$$

From above expressions, we can see that both updates are equivalent at the first iteration since $\boldsymbol{\mu}_1 = \mathbf{A}_1 \mathbf{m}_1 + \mathbf{c}$. Similarly, we can show that both updates are equivalent at every iteration by induction.

Thus, we can see that full-matrix square-root-free method is affine invariance. For a diagonal square-root-free method, it only preserves a reparametrization invariance with $\mathbf{A}$ being diagonal. Likewise, we can show that the update is affine invariance in a scaled case when using our empirical Fisher.

D. Derivation of Our Inverse-free and Square-root-free Full-AdaGrad

We show that we can make adaptive methods inverse-free via re-parameterization. This is useful to enables matrix methods to work in low-precision settings.

We consider performing NGD on another parametrization $\boldsymbol{\eta} \coloneqq (\boldsymbol{\mu}, \mathbf{S}^{-1})$ of the Gaussian, where $\mathbf{S}^{-1} \succ 0$ is known as the covariance matrix and is also positive-definite. We can perform NGD under this parameterization as

$$\begin{bmatrix}\boldsymbol{\mu} \\ \mathbf{S}^{-1}\end{bmatrix} \leftarrow \begin{bmatrix}\boldsymbol{\mu} \\ \mathbf{S}^{-1}\end{bmatrix} - \beta_1 \begin{bmatrix}\mathbf{S} & \mathbf{0} \\ \mathbf{0} & -\frac{1}{2}\frac{\partial \mathbf{S}}{\partial \mathbf{S}^{-1}}\end{bmatrix}^{-1} \begin{bmatrix}\partial_\mu \mathcal{L} \\ \partial_{S^{-1}} \mathcal{L}\end{bmatrix} = \begin{bmatrix}\boldsymbol{\mu} - \beta_1 \mathbf{S}^{-1}\partial_\mu \mathcal{L} \\ \mathbf{S}^{-1} + 2\beta_1 \partial_S \mathcal{L}\end{bmatrix} = \begin{bmatrix}\boldsymbol{\mu} - \beta_1 \mathbf{S}^{-1}\partial_\mu \mathcal{L} \\ \mathbf{S}^{-1} - 2\beta_1 \mathbf{S}^{-1}[\partial_{S^{-1}}\mathcal{L}]\mathbf{S}^{-1}\end{bmatrix},$$

where $\mathbf{F}_{\text{gauss}}(\boldsymbol{\eta}) = \begin{bmatrix}\mathbf{S} & \mathbf{0} \\ \mathbf{0} & -\frac{1}{2}\frac{\partial \mathbf{S}}{\partial \mathbf{S}^{-1}}\end{bmatrix}$ is the FIM of the Gaussian and is block-diagonal under this parameterization $\boldsymbol{\eta}$, and we use this gradient identity from matrix calculus $\partial_S \mathcal{L} = -\mathbf{S}^{-1}[\partial_{S^{-1}}\mathcal{L}]\mathbf{S}^{-1}$ in the last step.

In the above update, we use the old $\mathbf{S}^{-1}$ as a preconditioner. We can use the newly updated $\mathbf{S}_{\text{new}}^{-1}$ as a preconditioner as shown in the follwoing update rule

$$\mathbf{S}_{\text{new}}^{-1} \leftarrow (1-\beta_1\gamma)\mathbf{S}^{-1} - \beta_1 \mathbf{S}^{-1}\nabla_\mu^2 \ell(\boldsymbol{\mu})\mathbf{S}^{-1}, \;\; \boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \beta_1 \mathbf{S}_{\text{new}}^{-1}\nabla_\mu \ell(\boldsymbol{\mu}), \tag{24}$$

when performing NGD on parameterization $(\mathbf{S}\boldsymbol{\mu}, \mathbf{S}^{-1})$.

This above update is inverse-free since we directly update $\mathbf{S}^{-1}$ without inverting $\mathbf{S}$. To obtain inverse-free adaptive methods, we use the second moment $\nabla_\mu^2 \ell(\boldsymbol{\mu}) \approx \mathbf{H}$ as a Hessian approximation and introduce an additional learning rate β_2 to compensate this approximation.

However, the update of $\mathbf{S}^{-1}$ in (24) does not guarantee that $\mathbf{S}_{\text{new}}^{-1}$ is positive-definite even when $\mathcal{H} = \mathbf{H} = \mathbf{g}\mathbf{g}^\top$ is semi-positive-definite. Inspired by Lin et al. (2020), we propose to add a correction term highlighted in red in the update of $\mathbf{S}^{-1}$ to satisfy the constraint as

$$\mathbf{S}_{\text{new}}^{-1} \leftarrow (1-\beta_2\gamma)\mathbf{S}^{-1} - \beta_2 \mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1} + \frac{\beta_2^2}{2}\mathbf{W}, \tag{25}$$

where $\mathbf{W} \coloneqq \mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1} + 2\gamma \mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1} + \gamma^2 \mathbf{S}^{-1}$.

By setting $\gamma = 0$, we obtain an inverse-free update for square-root-free full-AdaGrad in Eq. (18) as

$$\mathbf{S}_{\text{new}}^{-1} \leftarrow \mathbf{S}^{-1} - \beta_2 \mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1} + \frac{\beta_2^2}{2}\mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1}, \;\; \boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \beta_1 \mathbf{S}_{\text{new}}^{-1}\mathbf{g}.$$

Claim 3. The update in Eq. (18) is guaranteed to be positive-definite when the current $\mathbf{S}^{-1}$ is positive-definite.

Proof. We prove that the update in (25) is positive-definite. Therefore, the update in Eq. (18) is also positive-definite by setting $\gamma = 0$.

When the current/initial $\mathbf{S}^{-1}$ is positive-definite, the updated $\mathbf{S}_{\text{new}}^{-1}$ in (25) is guaranteed to be positive-definite since

$$(1-\beta_2\gamma)\mathbf{S}^{-1} - \beta_2\mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1} + \frac{\beta_2^2}{2}\mathbf{W} = \mathbf{A}(\mathbf{I} - \beta_2\mathbf{N} + \frac{\beta_2^2}{2}\mathbf{N}^2)\mathbf{A}^\top = \tfrac{1}{2}\mathbf{A}\big[\mathbf{I} + (\mathbf{I}-\beta_2\mathbf{N})(\mathbf{I}-\beta_2\mathbf{N})^\top\big]\mathbf{A}^\top \succ 0, \tag{26}$$

where the current $\mathbf{S}^{-1}$ can be decomposed as $\mathbf{S}^{-1} = \mathbf{A}\mathbf{A}^\top$, $\mathbf{A}$ is a square non-singular matrix, and $\mathbf{N} \coloneqq \mathbf{A}^\top\mathbf{H}\mathbf{A} + \gamma\mathbf{I}$ is a symmetric matrix. □

E. Preconditioner Invariance for Full-matrix Square-root-free Adaptive Methods

We will show that following claim to explicitly demonstrate the preconditioner invariance. This invariance allows us to reparametrize a preconditioner $\mathbf{S}$ to obtain an inverse-free update scheme. In other words, the updates of $\bar{\mathbf{S}}^{-1}$ and $\mathbf{S}$ are equivalent up to a first-order accuracy thanks to this invariance.

Claim 4. Let $\bar{\mathbf{S}}^{-1}$ be the inverse of a preconditioner updated according to the inverse-free scheme (Eq. (18)) with this initialization $\bar{\mathbf{S}}_0^{-1} = \mathbf{S}_0^{-1}$. If $\bar{\mathbf{S}}$ and $\mathbf{S}$ are updated by using the same sequence of curvature approximations $\mathbf{H}$, then $\bar{\mathbf{S}}$ has a first-order accuracy of the root-free full-AdaGrad update of $\mathbf{S}$ (Eq. (17)) at each iteration, i.e., $\bar{\mathbf{S}} = \mathbf{S} + O(\beta_2^2)$.

Proof. We will show an equivalent relationship as $\bar{\mathbf{S}}_t^{-1}\mathbf{S}_t = \mathbf{I} + O(\beta_2^2)$ by induction. By the initialization, we know that this relationship holds when $t = 0$. Suppose that this relationship holds when $t = k-1$ so that $\bar{\mathbf{S}}_{k-1}^{-1}\mathbf{S}_{k-1} = \mathbf{I} + O(\beta_2^2)$.

Now, we show that $\bar{\mathbf{S}}_k^{-1}\mathbf{S}_k = \mathbf{I} + O(\beta_2^2)$. According to (18), $\bar{\mathbf{S}}^{-1}$ is updated as

$$\bar{\mathbf{S}}_k^{-1} = \bar{\mathbf{S}}_{k-1}^{-1} - \beta_2\bar{\mathbf{S}}_{k-1}^{-1}\mathbf{H}_{k-1}\bar{\mathbf{S}}_{k-1}^{-1} + \frac{\beta_2^2}{2}\bar{\mathbf{S}}_{k-1}^{-1}\mathbf{H}_{k-1}\bar{\mathbf{S}}_{k-1}^{-1}\mathbf{H}_{k-1}\bar{\mathbf{S}}_{k-1}^{-1}$$

According to Eq. (17), the root-free full-AdaGrad update of $\mathbf{S}$ is

$$\mathbf{S}_k = \mathbf{S}_{k-1} + \beta_2\mathbf{H}_{k-1}.$$

Thus, we have

$$\begin{aligned}
\bar{\mathbf{S}}_k^{-1}\mathbf{S}_k &= (\bar{\mathbf{S}}_{k-1}^{-1} - \beta_2\bar{\mathbf{S}}_{k-1}^{-1}\mathbf{H}_{k-1}\bar{\mathbf{S}}_{k-1}^{-1} + \frac{\beta_2^2}{2}\bar{\mathbf{S}}_{k-1}^{-1}\mathbf{H}_{k-1}\bar{\mathbf{S}}_{k-1}^{-1}\mathbf{H}_{k-1}\bar{\mathbf{S}}_{k-1}^{-1})(\mathbf{S}_{k-1} + \beta_2\mathbf{H}_{k-1}) \\
&= [\bar{\mathbf{S}}_{k-1}^{-1} - \beta_2\bar{\mathbf{S}}_{k-1}^{-1}\mathbf{H}_{k-1}\bar{\mathbf{S}}_{k-1}^{-1} + O(\beta_2^2)](\mathbf{S}_{k-1} + \beta_2\mathbf{H}_{k-1}) \\
&= \underbrace{\bar{\mathbf{S}}_{k-1}^{-1}\mathbf{S}_{k-1}}_{\mathbf{I}+O(\beta_2^2)} - \beta_2\bar{\mathbf{S}}_{k-1}^{-1}\mathbf{H}_{k-1}\underbrace{\bar{\mathbf{S}}_{k-1}^{-1}\mathbf{S}_{k-1}}_{\mathbf{I}+O(\beta_2^2)} + \beta_2\bar{\mathbf{S}}_{k-1}^{-1}\mathbf{H}_{k-1} + O(\beta_2^2) \\
&= \mathbf{I} - \beta_2\bar{\mathbf{S}}_{k-1}^{-1}\mathbf{H}_{k-1} + \beta_2\bar{\mathbf{S}}_{k-1}^{-1}\mathbf{H}_{k-1} + O(\beta_2^2) \\
&= \mathbf{I} + O(\beta_2)
\end{aligned}$$

Thus, by induction, we have $\bar{\mathbf{S}}_t^{-1}\mathbf{S}_t = \mathbf{I} + O(\beta_2^2)$. □

As shown in Lin et al. (2023b), the update of $\mathbf{A}$ is

$$\mathbf{A} = \mathbf{A}\exp(-\frac{\beta_2}{2}(\mathbf{A}^\top\mathbf{H}\mathbf{A} - \gamma\mathbf{I})) \tag{27}$$

where $(\mathbf{A}\mathbf{A}^\top)^{-1}$ is used to approximate a preconditioner.

Claim 5. Let $(\mathbf{A}\mathbf{A}^\top)^{-1}$ be a preconditioner, where $\mathbf{A}$ is initialized so that $(\mathbf{A}_0\mathbf{A}_0^\top)^{-1} = \mathbf{S}_0$. If $\mathbf{A}$ is updated according to the inverse-free scheme (Eq. (27) with $\gamma = 0$), and $\mathbf{A}$ and $\mathbf{S}$ are updated by using the same sequence of curvature approximations $\mathbf{H}$, then $(\mathbf{A}\mathbf{A}^\top)^{-1}$ has a first-order accuracy of the root-free full-AdaGrad update of $\mathbf{S}$ (Eq. (17)) at each iteration, i.e., $(\mathbf{A}\mathbf{A}^\top)^{-1} = \mathbf{S} + O(\beta_2^2)$.

Proof. We will show an equivalent relationship as $\mathbf{A}_t\mathbf{A}_t^\top\mathbf{S}_t = \mathbf{I} + O(\beta_2^2)$ by induction. By the initialization, we know that this relationship holds when $t = 0$. Suppose that this relationship holds when $t = k-1$ so that $\mathbf{A}_{k-1}\mathbf{A}_{k-1}^\top\mathbf{S}_{k-1} = \mathbf{I} + O(\beta_2^2)$.

Now, we show that $\mathbf{A}_k\mathbf{A}_k^\top\mathbf{S}_k = \mathbf{I} + O(\beta_2^2)$. According to the above update, $\mathbf{A}$ is updated as

$$\mathbf{A}_k = \mathbf{A}_{k-1}\exp(-\frac{\beta_2}{2}\mathbf{A}_{k-1}^\top\mathbf{H}_{k-1}\mathbf{A}_{k-1}) \tag{28}$$

and the product is

$$\begin{aligned}\mathbf{A}_k\mathbf{A}_k^\top &= \mathbf{A}_{k-1}\exp\big(-\beta_2\mathbf{A}_{k-1}^\top\mathbf{H}_{k-1}\mathbf{A}_{k-1}\big)\mathbf{A}_{k-1}^\top \\ &= \mathbf{A}_{k-1}\big(\mathbf{I} - \beta_2\mathbf{A}_{k-1}^\top\mathbf{H}_{k-1}\mathbf{A}_{k-1} + O(\beta_2^2)\big)\mathbf{A}_{k-1}^\top \\ &= \mathbf{A}_{k-1}\mathbf{A}_{k-1}^\top - \beta_2\mathbf{A}_{k-1}\mathbf{A}_{k-1}^\top\mathbf{H}_{k-1}\mathbf{A}_{k-1}\mathbf{C}_{k-1}^\top + O(\beta_2^2)\end{aligned}$$

Thus, we have

$$\begin{aligned}\mathbf{A}_k\mathbf{A}_k^\top\mathbf{S}_k &= \big[\mathbf{A}_{k-1}\mathbf{A}_{k-1}^\top - \beta_2\mathbf{A}_{k-1}\mathbf{A}_{k-1}^\top\mathbf{H}_{k-1}\mathbf{A}_{k-1}\mathbf{A}_{k-1}^\top + O(\beta_2^2)\big]\big(\mathbf{S}_{k-1} + \beta_2\mathbf{H}_{k-1}\big) \\ &= \underbrace{\mathbf{A}_{k-1}\mathbf{A}_{k-1}^\top\mathbf{S}_{k-1}}_{\mathbf{I}+O(\beta_2^2)} - \beta_2\mathbf{A}_{k-1}\mathbf{A}_{k-1}^\top\mathbf{H}_{k-1}\underbrace{\mathbf{A}_{k-1}\mathbf{A}_{k-1}^\top\mathbf{S}_{k-1}}_{\mathbf{I}+O(\beta_2^2)} + \beta_2\mathbf{A}_{k-1}\mathbf{A}_{k-1}^\top\mathbf{H}_{k-1} + O(\beta_2^2) \\ &= \mathbf{I} + O(\beta_2^2)\end{aligned}$$

Thus, by induction, we have $\mathbf{A}_t\mathbf{A}_t^\top\mathbf{S}_t = \mathbf{I} + O(\beta_2^2)$. □

F. Scale-invariance and Affine-invariance of Our Inverse-free Updates

Claim 6. Our root-free update shown in Eq. (18) is invariant to a scale transformation of the loss.

Proof. Recall that the (unscaled) optimization problem in (1) is

$$\min_{\mu} \ell(\boldsymbol{\mu}) \tag{29}$$

The scaled problem is

$$\min_{\bar{\mu}} \ell_{\text{scaled}}(\bar{\boldsymbol{\mu}}) \coloneqq \frac{1}{B}\ell(\bar{\boldsymbol{\mu}}) \tag{30}$$

We will show that our update is invariant at each step. In other words, $\boldsymbol{\mu} = \bar{\boldsymbol{\mu}}$ holds for every iteration. We assume $\boldsymbol{\mu} = \bar{\boldsymbol{\mu}}$ holds at the initialization step.

Recall that our update rule (Eq. (18)) to solve the original problem is

$$\begin{aligned}\mathbf{S}^{-1} &= \mathbf{S}^{-1} - \beta_2\mathbf{S}^{-1}\tilde{\mathbf{F}}_{\text{new}}\mathbf{S}^{-1} + \frac{\beta_2^2}{2}\mathbf{S}^{-1}\tilde{\mathbf{F}}_{\text{new}}\mathbf{S}^{-1}\tilde{\mathbf{F}}_{\text{new}}\mathbf{S}^{-1} \\ \boldsymbol{\mu} &= \boldsymbol{\mu} - \beta_1\mathbf{S}^{-1}\mathbf{g}\end{aligned}$$

where $\mathbf{g} \coloneqq \nabla_\mu\ell(\boldsymbol{\mu})$ and $\tilde{\mathbf{F}}_{\text{new}} = \mathbf{H} = \mathbf{g}\mathbf{g}^\top$. Recall that the gradient outer product $\mathbf{H}$ coincides with our empirical Fisher $\tilde{\mathbf{F}}_{\text{new}}$ in this case.

In the scaled case, our update is

$$\bar{\mathbf{S}}^{-1} = \bar{\mathbf{S}}^{-1} - \beta_2 \bar{\mathbf{S}}^{-1}\tilde{\mathbf{F}}_{\text{scaled}}\bar{\mathbf{S}}^{-1} + \frac{\beta_2^2}{2}\bar{\mathbf{S}}^{-1}\tilde{\mathbf{F}}_{\text{scaled}}\bar{\mathbf{S}}^{-1}\tilde{\mathbf{F}}_{\text{scaled}}\bar{\mathbf{S}}^{-1}$$
$$\bar{\boldsymbol{\mu}} = \bar{\boldsymbol{\mu}} - \beta_1 \bar{\mathbf{S}}^{-1}\bar{\mathbf{g}}$$

where $\bar{\mathbf{g}} \coloneqq \nabla_{\bar{\mu}}\ell_{\text{scaled}}(\bar{\boldsymbol{\mu}}) = \frac{1}{B}\mathbf{g}$ and $\tilde{\mathbf{F}}_{\text{new}} = B\bar{\mathbf{H}} = B\bar{\mathbf{g}}\bar{\mathbf{g}}^\top = \frac{1}{B}\mathbf{H}$

We first assume that $\bar{\mathbf{S}}^{-1} = B\mathbf{S}^{-1}$. Therefore, we can show that $\boldsymbol{\mu} = \bar{\boldsymbol{\mu}}$ since their descent directions are the same as $\bar{\mathbf{S}}^{-1}\bar{\mathbf{g}} = \big(B\mathbf{S}^{-1}\big)\big(\frac{1}{B}\mathbf{g}\big) = \mathbf{S}^{-1}\mathbf{g}$.

Now, we show that $\bar{\mathbf{S}} = \frac{\mathbf{S}}{B}$ at every iteration. We assume $\bar{\mathbf{S}}$ is initialized so that this relationship holds in the base case. Note that we have the following relationship

$$\begin{aligned}
\bar{\mathbf{S}}_{\text{new}}^{-1} &= \underbrace{\bar{\mathbf{S}}^{-1}}_{B\mathbf{S}^{-1}} - \beta_2\bar{\mathbf{S}}^{-1}\tilde{\mathbf{F}}_{\text{scaled}}\bar{\mathbf{S}}^{-1} + \frac{\beta_2^2}{2}\bar{\mathbf{S}}^{-1}\tilde{\mathbf{F}}_{\text{scaled}}\bar{\mathbf{S}}^{-1}\tilde{\mathbf{F}}_{\text{scaled}}\bar{\mathbf{S}}^{-1} \\
&= B\mathbf{S}^{-1} - \beta_2\big(B\mathbf{S}^{-1}\big)\frac{\mathbf{H}}{B}\big(B\mathbf{S}^{-1}\big) + \frac{\beta_2^2}{2}\big(B\mathbf{S}^{-1}\big)\frac{\mathbf{H}}{B}\big(B\mathbf{S}^{-1}\big)\frac{\mathbf{H}}{B}\big(B\mathbf{S}^{-1}\big) \\
&= B\big(\mathbf{S}^{-1} - \beta_2\mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1} + \frac{\beta_2^2}{2}\mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1}\big) \\
&= B\mathbf{S}_{\text{new}}^{-1}
\end{aligned}$$

Thus, we can use induction to establish this relationship.

□

Claim 7. Our root-free update in Eq. (18) is invariant up to an affine transformation.

Proof. This proof is similar to the proof in Appx.C. Recall that the (unscaled) optimization problem in (1) is

$$\min_{\mu} \ell(\boldsymbol{\mu}) \tag{31}$$

For simplicity, we only consider the linear transformation without a translation. Now, consider reparametrizing $\boldsymbol{\mu}$ with a known non-singular matrix $\mathbf{A}$ as $\boldsymbol{\mu} = \mathbf{A}\bar{\boldsymbol{\mu}}$. In this case, the optimization problem becomes

$$\min_{\bar{\mu}} \ell^{\text{rep}}(\bar{\boldsymbol{\mu}}) \coloneqq \ell(\mathbf{A}\bar{\mu}) \tag{32}$$

Let $\mathbf{S}$ and $\bar{\mathbf{S}}$ be preconditioners for the first problem and the second problem, respectively. We will show that $\bar{\mathbf{S}}^{-1} = \mathbf{A}^{-1}\mathbf{S}^{-1}\mathbf{A}^{-T}$. Once this relationship is established, we can use a similar proof technique in Appx. C to complete the proof.

We can establish this relationship by induction. Note that this relationship holds in the base case thanks to our initialization.

For the first problem, $\mathbf{S}^{-1}$ is updated as

$$\mathbf{S}_{\text{new}}^{-1} = \mathbf{S}^{-1} - \beta_2\mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1} + \frac{\beta_2^2}{2}\mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1}$$

By induction, we know that $\bar{\mathbf{S}}^{-1} = \mathbf{A}^{-1}\mathbf{S}^{-1}\mathbf{A}^{-T}$ at the current iteration. At a new iteration, the preconditioner $\bar{\mathbf{S}}^{-1}$ for the reparametrized problem is updated as

$$\begin{aligned}
\bar{\mathbf{S}}_{\text{new}}^{-1} &= \underbrace{\bar{\mathbf{S}}^{-1}}_{\mathbf{A}^{-1}\mathbf{S}^{-1}\mathbf{A}^{-T}} - \beta_2\bar{\mathbf{S}}^{-1}\underbrace{\bar{\mathbf{H}}}_{\mathbf{A}^\top\mathbf{H}\mathbf{A}}\bar{\mathbf{S}}^{-1} + \frac{\beta_2^2}{2}\bar{\mathbf{S}}^{-1}\bar{\mathbf{H}}\bar{\mathbf{S}}^{-1}\bar{\mathbf{H}}\bar{\mathbf{S}}^{-1} \\
&= \mathbf{A}^{-1}\mathbf{S}^{-1}\mathbf{A}^{-T} - \beta_2\mathbf{A}^{-1}\mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1}\mathbf{A}^{-T} + \frac{\beta_2^2}{2}\mathbf{A}^{-1}\mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1}\mathbf{A}^{-T} \\
&= \mathbf{A}^{-1}\big[\mathbf{S}^{-1} - \beta_2\mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1} + \frac{\beta_2^2}{2}\mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1}\mathbf{H}\mathbf{S}^{-1}\big]\mathbf{A}^{-T} \\
&= \mathbf{A}^{-1}\mathbf{S}_{\text{new}}^{-1}\mathbf{A}^{-T}
\end{aligned}$$

where we use the following identities.

$$\bar{\mathbf{g}} \coloneqq \nabla_{\bar{\mu}} \ell^{\text{rep}}(\bar{\boldsymbol{\mu}}) = \mathbf{A}^\top \mathbf{g}$$
$$\bar{\mathbf{H}} \coloneqq \bar{\mathbf{g}}\bar{\mathbf{g}}^\top = \mathbf{A}^\top \mathbf{H} \mathbf{A}$$

Thus, by induction, this relationship holds for every iteration.

□

We can similarly show that another inverse-free update (Eq. (27)) is both scale and affine invariant.

G. Derivation of Our Kronecker-factored Inverse-free Methods

We consider reparametrize the inverse preconditioner (covariance) as $\mathbf{S}^{-1} = (\mathbf{C}\mathbf{C}^\top) \otimes (\mathbf{K}\mathbf{K}^\top)$. To obtain an inverse-free update scheme, we consider directly update $\mathbf{C}$ and $\mathbf{K}$ to bypass the need for matrix inversion, where $\mathbf{C} \in \mathcal{R}^{p\times p}$ and $\mathbf{K} \in \mathcal{R}^{d\times d}$ are square non-singular matrices. Recall that we consider the curvature approximation $\mathcal{H} = \mathbf{H} = \mathbf{g}\mathbf{g}^\top$ as a term of the partial derivative as $2\partial_{S^{-1}}\mathcal{L} \approx \mathbf{H} - \gamma\mathbf{S} = \mathbf{g}\mathbf{g}^\top - \gamma(\mathbf{C}\mathbf{C}^\top)^{-1} \otimes (\mathbf{K}\mathbf{K}^\top)^{-1}$. We consider the local coordinates proposed by Lin et al. (2023b). For simplicity, we only update the block $\mathbf{C}$ while keeping blocks $\boldsymbol{\mu}$ and $\mathbf{K}$ frozen. Indeed, the blocks can be updated simultaneously. To update the block $\mathbf{C}$ at iteration t, the local coordinate $\boldsymbol{\eta}_C$ associated to $\mathbf{C}_t$ is defined as $\mathbf{S}^{-1} = (\mathbf{C}\mathbf{C}^\top) \otimes (\mathbf{K}\mathbf{K}^\top)$, where $\mathbf{C} = \mathbf{C}_t\text{Exp}(\frac{\boldsymbol{\eta}_C}{\sqrt{2d}})$. $\boldsymbol{\eta}_C$ is a square symmetric matrix and it can be singular. Lin et al. (2023b) show that the block approximated FIM of the Gaussian can be orthonormal at the origin in this local coordinate system as $\mathbf{F}_{\text{gauss}}(\boldsymbol{\eta}_C^{\text{cur}}) = \mathbf{I}$. Importantly, the origin in this system represents the current block $\mathbf{C}_t$ as $\mathbf{C}_t \equiv \mathbf{C}_t\text{Exp}(\frac{\boldsymbol{\eta}_C^{\text{cur}}}{\sqrt{2d}})$ where $\boldsymbol{\eta}_C^{\text{cur}} = \mathbf{0}$. Thus, we can perform NGD in this local coordinate system as

$$\boldsymbol{\eta}_C^{\text{new}} \leftarrow \boldsymbol{\eta}_C^{\text{cur}} - \beta_2 \left(\mathbf{F}_{\text{gauss}}(\boldsymbol{\eta}_C^{\text{cur}})\right)^{-1} \partial_{\eta_C}\mathcal{L} = \mathbf{0} - \beta_2 \partial_{\eta_C}\mathcal{L} \tag{33}$$
$$\mathbf{C}_{t+1} \leftarrow \mathbf{C}_t\text{Exp}(\frac{\boldsymbol{\eta}_C^{\text{new}}}{\sqrt{2d}}) \approx \mathbf{C}_t(\mathbf{I} + \frac{\boldsymbol{\eta}_C^{\text{new}}}{\sqrt{2d}}) \tag{34}$$

where we can use the chain rule to compute the partial derivative at $\boldsymbol{\eta}_C^{\text{cur}} = \mathbf{0}$ and $\mathcal{H} = \mathbf{H}$

$$2\partial_{\eta_C}\mathcal{L} = 2\frac{\partial \mathbf{S}^{-1}}{\partial \boldsymbol{\eta}_C}\partial_{S^{-1}}\mathcal{L} \approx \frac{\partial \mathbf{S}^{-1}}{\partial \boldsymbol{\eta}_C}\left[\mathbf{g}\mathbf{g}^\top - \gamma(\mathbf{C}_t\mathbf{C}_t^\top)^{-1} \otimes (\mathbf{K}\mathbf{K}^\top)^{-1}\right] = \sqrt{\frac{2}{d}}\mathbf{C}^\top\mathbf{G}^\top\mathbf{K}\mathbf{K}^\top\mathbf{G}\mathbf{C} - \gamma\sqrt{2d}\mathbf{I}_p. \tag{35}$$

Thus, the update for block $\mathbf{C}$ can be re-expressed as

$$\mathbf{C}_{t+1} \leftarrow \mathbf{C}_t\text{Exp}(-\frac{\beta_2}{2d}\big(\mathbf{C}^\top\mathbf{G}^\top\mathbf{K}\mathbf{K}^\top\mathbf{G}\mathbf{C} - \gamma d\mathbf{I}_p\big)) \approx \mathbf{C}_t\big[\mathbf{I} - \frac{\beta_2}{2d}\big(\mathbf{C}^\top\mathbf{G}^\top\mathbf{K}\mathbf{K}^\top\mathbf{G}\mathbf{C} - \gamma d\mathbf{I}_p\big)\big] \tag{36}$$

We also include a damping term $\lambda\mathbf{I}_{dp} = \lambda\mathbf{I}_d \otimes \mathbf{I}_p$ into the curvature approximation such as $\mathcal{H} = \mathbf{g}\mathbf{g}^\top + \lambda\mathbf{I}_d \otimes \mathbf{I}_p$. Recall that we do not assume that the curvature approximation $\mathcal{H}$ has the same structure as the preconditioner $\mathbf{S}$. We can similarly update blocks $\mathbf{K}$ and $\boldsymbol{\mu}$. The deails of the complete update can be found at Fig. 7.

H. Scale-invariance and Affine-invariance of Our Kronecker-factored Updates

Claim 8. Our root-free Shampoo update shown in Fig. 5 is invariant to a scale-transformation of the loss.

Proof. We consider a matrix weight $\boldsymbol{\mu} = \text{vec}(\mathbf{M})$. In this case, the (unscaled) optimization problem in (1) is

$$\min_M \ell(\mathbf{M}) \tag{37}$$

where $\mathbf{M} \in \mathbb{R}^{p\times d}$.

The scaled problem is

$$\min_{\bar{M}} \ell_{\text{scaled}}(\bar{\mathbf{M}}) \coloneqq \frac{1}{B}\ell(\bar{\mathbf{M}}) \tag{38}$$

We will show that our update is invariant at each step. In other words, $\mathbf{M} = \bar{\mathbf{M}}$ holds for every iteration. We assume $\mathbf{M} = \bar{\mathbf{M}}$ holds at the initialization step.

Recall that our update rule (Fig. 5) to solve the original problem is

$$\begin{aligned}\mathbf{S}_C &= (1-\beta_2\gamma)\mathbf{S}_C + \frac{\beta_2}{d}(\mathbf{G}\mathbf{S}_K^{-1}\mathbf{G}^\top)\\ \mathbf{S}_K &= (1-\beta_2\gamma)\mathbf{S}_K + \frac{\beta_2}{p}(\mathbf{G}^\top\mathbf{S}_C^{-1}\mathbf{G})\\ \mathbf{M} &= \mathbf{M} - \beta_1\mathbf{S}_C^{-1}\mathbf{G}\mathbf{S}_K^{-1}\end{aligned}$$

where $\mathbf{G} \coloneqq \nabla_M \ell(\mathbf{M}) \in \mathbb{R}^{p\times d}$.

In this scaled case, our update shown in Fig. 5 is

$$\begin{aligned}\bar{\mathbf{S}}_C &= (1-\beta_2\gamma)\bar{\mathbf{S}}_C + \frac{\beta_2}{d}(B\bar{\mathbf{G}}\bar{\mathbf{S}}_K^{-1}\bar{\mathbf{G}}^\top)\\ \bar{\mathbf{S}}_K &= (1-\beta_2\gamma)\bar{\mathbf{S}}_K + \frac{\beta_2}{p}(B\bar{\mathbf{G}}^\top\bar{\mathbf{S}}_C^{-1}\bar{\mathbf{G}})\\ \bar{\mathbf{M}} &= \bar{\mathbf{M}} - \beta_1\bar{\mathbf{S}}_C^{-1}\bar{\mathbf{G}}\bar{\mathbf{S}}_K^{-1}\end{aligned}$$

where $\bar{\mathbf{G}} = \nabla_{\bar{M}}\ell_{\text{scaled}}(\bar{\mathbf{M}}) = \frac{1}{B}\mathbf{G}$.

We first assume that $\bar{\mathbf{S}}_C = \frac{\mathbf{S}_C}{\sqrt{B}}$ and $\bar{\mathbf{S}}_K = \frac{\mathbf{S}_K}{\sqrt{B}}$. Therefore, we can show that $\mathbf{M} = \bar{\mathbf{M}}$ since their descent directions are the same as $\bar{\mathbf{S}}_C^{-1}\bar{\mathbf{G}}\bar{\mathbf{S}}_K^{-1} = \big(\frac{\mathbf{S}_C}{\sqrt{B}}\big)^{-1}\big(\frac{\mathbf{G}}{B}\big)\big(\frac{\mathbf{S}_K}{\sqrt{B}}\big)^{-1} = \mathbf{S}_C^{-1}\mathbf{G}\mathbf{S}_K^{-1}$.

Now, we prove that $\bar{\mathbf{S}}_C = \frac{\mathbf{S}_C}{\sqrt{B}}$ and $\bar{\mathbf{S}}_K = \frac{\mathbf{S}_K}{\sqrt{B}}$. We will prove this by induction. We assume that $\bar{\mathbf{S}}_C$ and $\bar{\mathbf{S}}_K$ are initialized so that $\bar{\mathbf{S}}_C = \frac{\mathbf{S}_C}{\sqrt{B}}$ and $\bar{\mathbf{S}}_K = \frac{\mathbf{S}_K}{\sqrt{B}}$ hold in the base case needed for induction. It is easy to see that

$$\begin{aligned}\bar{\mathbf{S}}_C^{\text{new}} &= (1-\beta_2\gamma)\bar{\mathbf{S}}_C + \frac{\beta_2}{d}(B\bar{\mathbf{G}}\bar{\mathbf{S}}_K^{-1}\bar{\mathbf{G}}^\top)\\ &= (1-\beta_2\gamma)\big(\frac{\mathbf{S}_C}{\sqrt{B}}\big) + \frac{\beta_2}{d}\Big(B\frac{\mathbf{G}}{B}\big(\frac{\mathbf{S}_K}{\sqrt{B}}\big)^{-1}\frac{\mathbf{G}}{B}^\top\Big)\\ &= \frac{1}{\sqrt{B}}\big[(1-\beta_2\gamma)\mathbf{S}_C + \frac{\beta_2}{d}(\mathbf{G}\mathbf{S}_K^{-1}\mathbf{G}^\top)\big]\\ &= \frac{\mathbf{S}_C^{\text{new}}}{\sqrt{B}}\end{aligned}$$

Thus, by induction, we can show $\bar{\mathbf{S}}_C = \frac{\mathbf{S}_C}{\sqrt{B}}$. We can similarly show that $\bar{\mathbf{S}}_K = \frac{\mathbf{S}_K}{\sqrt{B}}$.

□

Claim 9. Our root-free Shampoo update shown in Fig. 5 is affine reparametrization invariant up to an (Kronecker-factored) affine transformation.

Proof. We consider the following unconstrained optimization problem with a matrix weight $\mathbf{M} \in \mathbb{R}^{p\times d}$.

$$\min_M \ell(\mathbf{M}) \tag{39}$$

Now, consider reparametrizing $\mathbf{M}$ with known non-singular transformation matrices $\mathbf{E} \in \mathbb{R}^{p\times p}$ and $\mathbf{F} \in \mathbb{R}^{d\times d}$ and a constant matrix $\mathbf{K}$ as $\mathbf{M} = \mathbf{E}\mathbf{N}\mathbf{F}^T + \mathbf{O}$ In this case, the optimization problem becomes

$$\min_N \ell^{\text{rep}}(\mathbf{N}) \coloneqq \ell(\mathbf{E}\mathbf{N}\mathbf{F}^\top + \mathbf{O}) \tag{40}$$

Note that this is a Kronecker-factored affine transformation when we a vector representation. In other words, $\mathrm{vec}(\mathbf{M}) = (\mathbf{F} \otimes \mathbf{E})\mathrm{vec}(\mathbf{N}) + \mathrm{vec}(\mathbf{O})$, where this transformation matrix $(\mathbf{F} \otimes \mathbf{E})$ admits a Kronecker-factored structure.

We will show that our update is affine invariant for such a transformation at each step. In other words, if we use the same update rule to solve these two problems, they are equivalent.

For the first problem, the method takes the following step at iteration t

$$\begin{aligned}
\mathbf{S}_C^{(t+1)} &= (1-\beta_2\gamma)\mathbf{S}_C^{(t)} + \frac{\beta_2}{d}\mathbf{G}^{(t)}\big(\mathbf{S}_K^{(t)}\big)^{-1}\big(\mathbf{G}^{(t)}\big)^\top,\\
\mathbf{S}_K^{(t+1)} &= (1-\beta_2\gamma)\mathbf{S}_K^{(t)} + \frac{\beta_2}{p}\big(\mathbf{G}^{(t)}\big)^\top\big(\mathbf{S}_C^{(t)}\big)^{-1}\mathbf{G}^{(t)},\\
\mathbf{M}^{(t+1)} &= \mathbf{M}^{(t)} - \beta_1\big(\mathbf{S}_C^{(t+1)}\big)^{-1}\mathbf{G}^{(t)}\big(\mathbf{S}_K^{(t+1)}\big)^{-1}
\end{aligned}$$

where $\mathbf{G}^{(t)} \coloneqq \nabla_M \ell(\mathbf{M})\big|_{M=M^{(t)}}$.

For the second problem, we assume $\mathbf{S}_{C^{\text{rep}}}$, $\mathbf{S}_{K^{\text{rep}}}$, and $\mathbf{N}$ are initialized so that $\mathbf{S}_{C^{\text{rep}}}^{(0)} = \mathbf{E}^{-T}\mathbf{S}_C^{(0)}\mathbf{E}^{-1}$, $\mathbf{S}_{K^{\text{rep}}}^{(0)} = \mathbf{F}^{-T}\mathbf{S}_K^{(0)}\mathbf{F}^{-1}$, and $\mathbf{E}^{-1}(\mathbf{M}^{(0)} - \mathbf{O})\mathbf{F}^{-T} = \mathbf{N}^{(0)}$.

In this case, the our update at the first iteration becomes

$$\begin{aligned}
\mathbf{S}_{C^{\text{rep}}}^{(t+1)} &= (1-\beta_2\gamma)\mathbf{S}_{C^{\text{rep}}}^{(t)} + \frac{\beta_2}{d}\mathbf{G}_{\text{rep}}^{(t)}\big(\mathbf{S}_{K^{\text{rep}}}^{(t)}\big)^{-1}\big(\mathbf{G}_{\text{rep}}^{(t)}\big)^\top,\\
\mathbf{S}_{K^{\text{rep}}}^{(t+1)} &= (1-\beta_2\gamma)\mathbf{S}_{K^{\text{rep}}}^{(t)} + \frac{\beta_2}{p}\big(\mathbf{G}_{\text{rep}}^{(t)}\big)^\top\big(\mathbf{S}_{C^{\text{rep}}}^{(t)}\big)^{-1}\mathbf{G}_{\text{rep}}^{(t)},\\
\mathbf{N}^{(t+1)} &= \mathbf{N}^{(t)} - \beta_1\big(\mathbf{S}_{C^{\text{rep}}}^{(t+1)}\big)^{-1}\mathbf{G}_{\text{rep}}^{(t)}\big(\mathbf{S}_{K^{\text{rep}}}^{(t+1)}\big)^{-1}
\end{aligned}$$

where we use the following identities when $t = 0$.

$$\mathbf{G}_{\text{rep}}^{(t)} \coloneqq \nabla_N \ell^{\text{rep}}(\mathbf{N})\big|_{N=N^{(t)}} = \frac{\partial \mathbf{M}}{\partial \mathbf{N}}|_{M=M^{(t)}} \nabla_M \ell^{\text{rep}}(\mathbf{M})\big|_{M=M^{(t)}} = \mathbf{E}^\top \nabla_M \ell(\mathbf{M})|_{M=M^{(t)}}\mathbf{F} = \mathbf{E}^\top \mathbf{G}^{(t)}\mathbf{F} \tag{41}$$

It is easy to see that the following expressions hold when $t = 0$.

$$\begin{aligned}
\mathbf{E}^{-T}\mathbf{S}_{C^{\text{rep}}}^{(t+1)}\mathbf{E}^{-1} &= (1-\beta_2\gamma)\mathbf{E}^{-T}\mathbf{S}_{C^{\text{rep}}}^{(t)}\mathbf{E}^{-1} + \frac{\beta_2}{d}\mathbf{E}^{-T}\underbrace{\mathbf{G}_{\text{rep}}^{(t)}}_{E^\top G^{(t)} F}\big(\mathbf{S}_{K^{\text{rep}}}^{(t)}\big)^{-1}\big(\mathbf{G}_{\text{rep}}^{(t)}\big)^\top\mathbf{E}^{-1},\\
&= (1-\beta_2\gamma)\underbrace{\mathbf{E}^{-T}\mathbf{S}_{C^{\text{rep}}}^{(t)}\mathbf{E}^{-1}}_{S_C^{(t)}} + \frac{\beta_2}{d}\mathbf{G}^{(t)}\big(\underbrace{\mathbf{F}^{-T}\mathbf{S}_{K^{\text{rep}}}^{(t)}\mathbf{F}^{-1}}_{S_K^{(t)}}\big)^{-1}\big(\mathbf{G}^{(t)}\big)^\top\\
&= (1-\beta_2\gamma)\mathbf{S}_C^{(t)} + \frac{\beta_2}{d}\mathbf{G}^{(t)}\big(\mathbf{S}_K^{(t)}\big)^{-1}\big(\mathbf{G}^{(t)}\big)^\top\\
&= \mathbf{S}_C^{(t+1)}
\end{aligned}$$

Similarly, we can show $\mathbf{F}^{-T}\mathbf{S}_{K^{\text{rep}}}^{(t+1)}\mathbf{F}^{-1} = \mathbf{S}_K^{(t+1)}$ when $t = 0$.

Consequently, we have the following result when $t = 0$

$$\begin{aligned}
\mathbf{E}\mathbf{N}^{(t+1)}\mathbf{F}^\top &= \mathbf{E}\mathbf{N}^{(t)}\mathbf{F}^\top - \beta_1\mathbf{E}\big(\mathbf{S}_{C^{\text{rep}}}^{(t+1)}\big)^{-1}\underbrace{\mathbf{G}_{\text{rep}}^{(t)}}_{E^\top G^{(t)} F}\big(\mathbf{S}_{K^{\text{rep}}}^{(t+1)}\big)^{-1}\mathbf{F}^\top\\
&= \underbrace{\mathbf{E}\mathbf{N}^{(t)}\mathbf{F}^\top}_{M^{(t)}-O} - \beta_1\underbrace{\mathbf{E}\big(\mathbf{S}_{C^{\text{rep}}}^{(t+1)}\big)^{-1}\mathbf{E}^\top}_{\left(S_C^{(t+1)}\right)^{-1}}\mathbf{G}^{(t)}\underbrace{\mathbf{F}\big(\mathbf{S}_{K^{\text{rep}}}^{(t+1)}\big)^{-1}\mathbf{F}^\top}_{\left(S_K^{(t+1)}\right)^{-1}} = \mathbf{M}^{(t+1)} - \mathbf{O}.
\end{aligned}$$

In other words, $\mathbf{M}^{(t+1)} = \mathbf{E}\mathbf{N}^{(t+1)}\mathbf{F}^\top + \mathbf{O}$ when $t = 0$. From this expression, we can see that both updates are equivalent at the first iteration. Similarly, we can show that both updates are equivalent at every iteration by induction.

□

We can also show that our inverse-free Shampoo update shown in Fig. 5 is both scale and affine invariant.

I. Preconditioner Invariance of Our Kornecker-factored Updates

We reparametrize a Kronecker-factored preconditioner $\mathbf{S} = \mathbf{S}_C \otimes \mathbf{S}_K$ to obtain an inverse-free update scheme. The following claim explicitly demonstrates the preconditioner invariance. In other words, the updates of $\mathbf{C}$ and $\mathbf{S}_C$ are equivalent up to a first-order accuracy.

Claim 10. Let $\mathbf{C}\mathbf{C}^\top$ be the inverse of a preconditioner updated according to the inverse-free scheme in Fig. 5 with initialization $\mathbf{C}\mathbf{C} = \mathbf{S}_C^{-1}$. If $\mathbf{C}$ and $\mathbf{S}_C$ are updated by using the same sequence of gradients $\mathbf{G}$, then $\mathbf{C}\mathbf{C}^\top$ has a first-order accuracy of our root-free Shampoo update of $\mathbf{S}_C$ in Fig. 5 at each iteration, i.e., $\mathbf{C}\mathbf{C}^\top = \mathbf{S}_C^{-1} + O(\beta_2^2)$. Similarly, $\mathbf{K}\mathbf{K}^\top$ has a first-order accuracy of our root-free update of $\mathbf{S}_K$ at each iteration, i.e., $\mathbf{K}\mathbf{K}^\top = \mathbf{S}_K^{-1} + O(\beta_2^2)$.

Proof. It is equivalent to show that $(\mathbf{C}\mathbf{C}^\top)\mathbf{S}_C = \mathbf{I} + O(\beta_2^2)$ and $(\mathbf{K}\mathbf{K}^\top)\mathbf{S}_K = \mathbf{I} + O(\beta_2^2)$. We will prove this by induction. Thanks to the initialization, we know that the base case ($t = 0$) is true. Now, we assume this relationship holds when $t = k$. Consider the case when $t = k + 1$.

For notation simplicity, we drop the index k and denote $\mathbf{C}^{(k+1)}$ and $\mathbf{S}_C^{(k+1)}$ by $\mathbf{C}^{\text{new}}$ and $\mathbf{S}_C^{\text{new}}$. We have the following result

$$
\begin{aligned}
&\big[\mathbf{C}^{\text{new}}\big(\mathbf{C}^{\text{new}}\big)^\top\big]\mathbf{S}_C^{\text{new}} \\
=&\mathbf{C}\big[\mathbf{I} - \frac{\beta_2}{d}\big(\mathbf{C}^\top\mathbf{G}\mathbf{K}\mathbf{K}^\top\mathbf{G}^\top\mathbf{C} - \gamma d\mathbf{I}\big) + O(\beta_2^2)\big]\mathbf{C}^\top\big[(1-\gamma\beta_2)\mathbf{S}_C + \frac{\beta_2}{d}\mathbf{G}\mathbf{S}_K^{-1}\mathbf{G}^\top\big] \\
=&\big[(1+\gamma\beta_2)\mathbf{C}\mathbf{C}^\top - \frac{\beta_2}{d}\mathbf{C}\mathbf{C}^\top\mathbf{G}\mathbf{K}\mathbf{K}^\top\mathbf{G}^\top\mathbf{C}\mathbf{C}^\top\big]\big[(1-\gamma\beta_2)\mathbf{S}_C + \frac{\beta_2}{d}\mathbf{G}\mathbf{S}_K^{-1}\mathbf{G}^\top\big] + O(\beta_2^2) \\
=&\underbrace{\mathbf{C}\mathbf{C}^\top\mathbf{S}_C}_{I+O(\beta_2^2)} - \frac{\beta_2}{d}\mathbf{C}\mathbf{C}^\top\mathbf{G}\mathbf{K}\mathbf{K}^\top\mathbf{G}^\top\underbrace{\mathbf{C}\mathbf{C}^\top\mathbf{S}_C}_{I+O(\beta_2^2)} + \frac{\beta_2}{d}\mathbf{C}\mathbf{C}^\top\mathbf{G}\underbrace{\mathbf{S}_K^{-1}}_{KK^\top+O(\beta_2^2)}\mathbf{G}^\top + O(\beta_2^2) \\
=&\mathbf{I} + O(\beta_2^2)
\end{aligned}
$$

Likewise, we have $\big[\mathbf{K}^{\text{new}}\big(\mathbf{K}^{\text{new}}\big)^\top\big]\mathbf{S}_K^{\text{new}} = \mathbf{I} + O(\beta_2^2)$. Thus, we can use induction to show the relationship holds at every iteration. □

As discussed in the caption of Fig.7, we can make the higher order term $O(\beta_2^2)$ negligible when β_2 is determined by the matrix norm of $\frac{1}{d}\big(\mathbf{C}^\top\mathbf{G}\mathbf{K}\mathbf{K}^\top\mathbf{G}^\top\mathbf{C} - \gamma d\mathbf{I}\big)$.

J. Experimental Details & Additional Experiments

We consider various NN models ranging from classical to modern models to demonstrate the effectiveness of root-free adaptive methods. We consider the following NN models in our (pre-)training experiments.

- CNNs: ResNet34, VGG16, DenseNet121 on CIFAR100
- RNN: 3-layer LSTM on PenTree
- GNN: Graph MLP with attention on OgbnProducts
- Transformers: SwinViT, FocalNet, GCViT on ImageWoof10
- Mamba: VMamba on ImageWoof10

We train CNNs for 210 epochs, and ViTs and VMamba for 300 epochs with mini-batch size 128. For CNNs, RNN, and GNN, we use a step decay schedule suggested by Wilson et al. (2017). For Transformers and Mamba, we use a cosine learning rate schedule suggested by Chen et al. (2023). We will release the code to reproduce our experiments.

For fine-tuning experiments, we use pre-trained weights of a ViT model (Dosovitskiy et al., 2020) and fine-tune them on CIFAR100 and FOOD101 datasets. We use this implementation (`https://github.com/bwconrad/vit-finetune`) in our experiments.

Hyperparameter Tuning We use PyTorch's built-in SGD, AdamW, and RMSProp. For Shampoo, we rely on the state-of-the-art PyTorch implementation from Meta (Shi et al., 2023). We tune the following hyperparameters (HPs) for each optimizer.

- SGD: initial learning rate, momentum, weight decay
- AdamW: initial learning rate, coefficients used for estimating the first and second moment, weight decay, damping
- RMSProp: initial learning rate, coefficient used for estimating the second moment, momentum, weight decay, damping
- Shampoo: initial learning rate, coefficients used for updating momentum and Kronecker factors, weight decay, damping
- RF-RMSProp (ours, c.f. Fig. 6): initial learning rate, coefficients used for estimating the second moment, momentum, weight decay, damping
- IF-Shampoo with $\gamma = 1$ (ours, c.f. Fig. 7): initial learning rate, coefficients used for updating momentum and Kronecker factors, weight decay, damping, Riemannian momentum

For matrix adaptive methods (Shampoo and IF-Shampoo), we update their matrix preconditioners at each 2 iterations. For all tasks and optimizers, we employ a two-stage HP tuning protocol based on random search (Choi et al., 2019). In the first stage, we use larger search regimes for all HPs. Based on this stage, we select a narrower HP range and re-run the search, reporting the best run for each method. Each stage use 100 runs and we will release the code to reproduce our experiments that contain HP search space details in the second-stage.

Mixed-precesion Training For all optimizers, only the forward pass is executed in mixed precision with BFP-16 (as recommended by the official PyTorch guide). The gradients are automatically cast back to FP-32 by PyTorch. Shampoo uses these FP-32 gradients for its preconditioner and is unstable when converting them to BFP-16 (Shi et al., 2023) . Instead, our IF-Shampoo converts the gradients into BFP-16, updates the preconditioner, and even takes preconditioned gradient steps in half precision. Our method works well in half precision without using matrix decomposition and matrix solve/inversion. These matrix operations in half precision are not supported in PyTorch and JAX because they are numerically unstable.

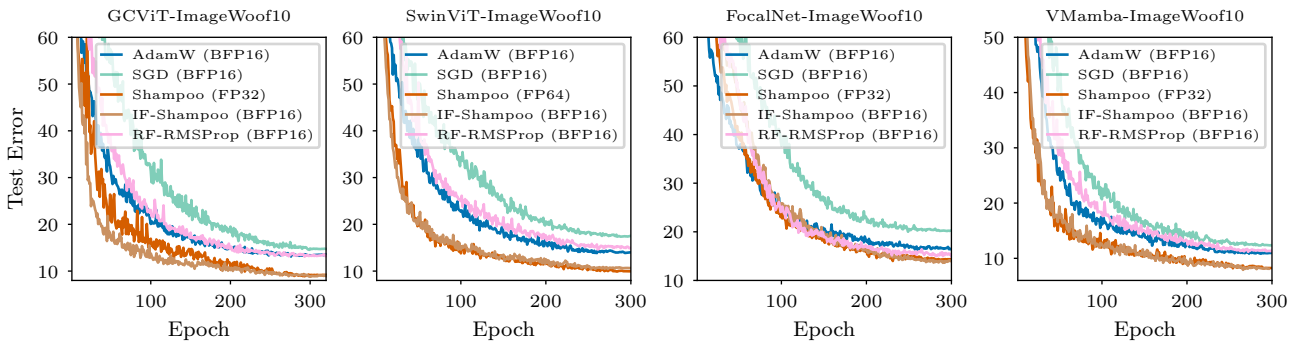

Figure 8. Comparison of matrix root-free versus root-based methods on GCViT (Hatamizadeh et al., 2023), SwinViT (Liu et al., 2021), FocalNet (Yang et al., 2022), and VMamba (Liu et al., 2024). Both matrix methods (Shampoo, IF-Shampoo) outperform diagonal methods on modern vision models such as transformers and mambas using modern training strategies (cosine learning rate schedule, random search using 200 runs). In contrast to Shampoo, our inverse-free matrix method, IF-Shampoo, runs in BFP-16 and trains at least twice as fast, while using less memory. We update matrix preconditioners at each 2 iterations and can further reduce the wall clock time by updating them less frequently.

Dataset	Method	GCViT	SwinViT	FocalNet	VMamba
ImageWoof10	AdamW	13.48/143	13.96/162	16.58/159	10.96/315
	SGD	14.74/142	17.44/161	20.22/158	12.38/314
	Shampoo	9.16/550	9.97/633	14.33/716	8.25/1336
	RF-RMSProp (ours)	13.37/144	15.02/162	15.39/159	11.38/315
	IF-Shampoo (ours)	9.05/202	10.65/169	13.93/189	8.20/384

Table 1. Results about the performance (test error/clock time) of the optimizers on modern NN models. We train all models for 300 epochs. All methods except Shampoo support training with BFP-16. Shampoo has to use FP-32— sometimes FP-64—to update its preconditioners to avoid numerical instabilities. For example, Shampoo has to use FP-64 on SwinViT. For matrix adaptive methods (Shampoo, IF-Shampoo), we update their preconditioners at every 2 iterations and can further reduce the clock time by updating them less frequently. The results are obtained by averaging over the last 10 iterations.

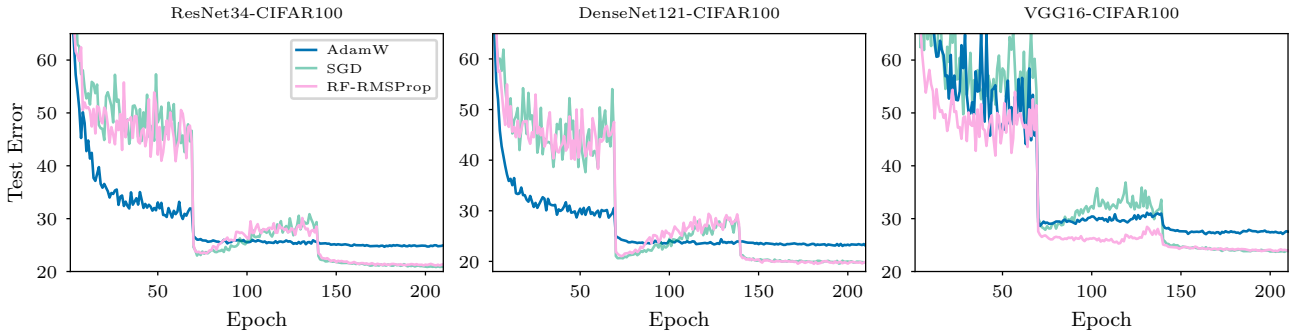

Figure 9. Comparison of root-free versus root-based adaptive methods on ResNet34, DenseNet121, and VGG16 using modern training strategies (step decay learning rate schedule, random search using 200 runs). Root-free adaptive methods close the generalization gap between their root-based counterparts and SGD on CNNs.

Dataset	Method	ResNet34	DenseNet121	VGG16
CIFAR100	AdamW	24.85	23.30	27.42
	SGD	20.91	19.88	24.02
	RF-RMSProp (ours)	21.25	19.81	24.00

Table 2. Results about the performance (test error) of the optimizers on convolutional NN models. The results are obtained by averaging over the last 10 iterations.

Additional Experiments We also evaluate our root-free methods on regression and fine-tuning problems (c.f. Fig. 10 and 11).

Figure 10. Experiments demonstrating that square-root-free adaptive methods work well with both cross entropy (CE) and square error (SE) losses. Thus, our (diagonal) empirical Fisher used in our update scheme does not suffer from the limitations of the standard empirical Fisher (Kunstner et al., 2019). We train all models using mini-batches and use random search (200 runs) to tune these methods. In the first two plots on the left, we consider convex problems using a constant learning rate schedule (outdated training scheme) considered by Kunstner et al. (2019). In the remaining plots, we consider non-convex NN problems with a step decay learning rate schedule (modern training scheme) considered by Wilson et al. (2017). We consider ResNet50 models and train them for 120 epochs with mini-batch size 128. Due to the large number of classes on the CIFAR100 dataset, we employ SE loss functions suggested by Hui & Belkin (2020) when it comes to using SE loss functions for classification tasks.

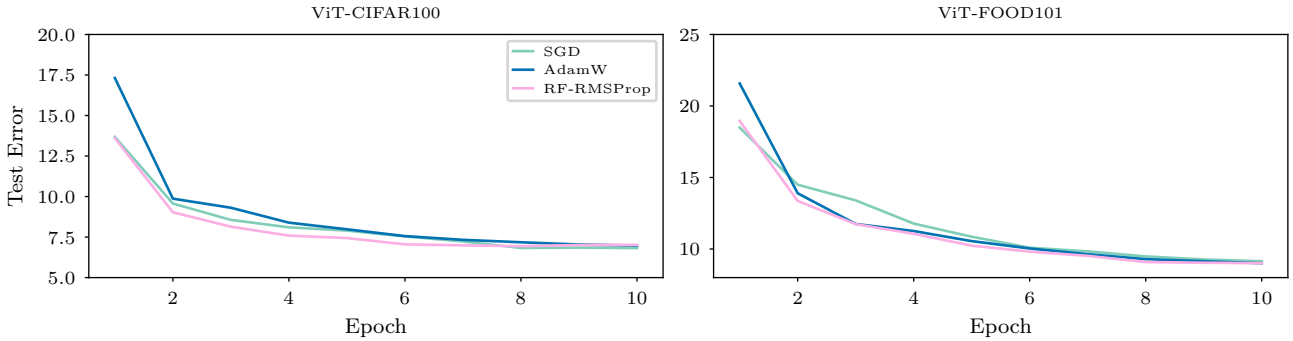

Figure 11. Experiments demonstrating that square-root-free adaptive methods work well in fine-tuning settings, where we use a ViT model (Dosovitskiy et al., 2020) pretrained on ImageNet-21k and fine-tune on CIFAR100 and FOOD101 datasets.