Faster Adaptive Optimization via Expected Gradient Outer Product Reparameterization

PENDING

Abstract. Adaptive optimization algorithms—such as Adagrad, Adam, and their variants—have found widespread use in machine learning, signal processing and many other settings. Several methods in this family are not rotationally equivariant, meaning that simple reparameterizations (i.e. change of basis) can drastically affect their convergence. However, their sensitivity to the choice of parameterization has not been systematically studied; it is not clear how to identify a "favorable" change of basis in which these methods perform best. In this paper we propose a reparameterization method and demonstrate both theoretically and empirically its potential to improve their convergence behavior. Our method is an orthonormal transformation based on the *expected gradient outer product* (EGOP) matrix, which can be approximated using either full-batch or stochastic gradient oracles. Our theoretical results show that in the neighborhoods of local minima, the sensitivity of adaptive algorithms to choice-of-basis is influenced by spectral decay in the EGOP matrix. We illustrate the potential impact of EGOP reparameterization by presenting empirical evidence and theoretical arguments that common machine learning tasks with "natural" data exhibit EGOP spectral decay.

Key words. optimization, adaptive gradient methods, Adam, Adagrad, expected gradient outer product, reparameterization, low-rank, preconditioning, neural networks

MSC codes. 68T07 (Computer science: artificial NN's and deep learning), 90C99 (Mathematical programming)

1. Introduction.

Adaptive optimization algorithms are popular methods in modern machine learning [8]. Optimizers in this family include the seminal Adagrad and Adam algorithms as well as variants such as AdamW, Adadelta, and Adamax [22, 26, 32, 41]. These algorithms are termed "adaptive" because they maintain and update different learning rates for each coordinate in parameter space.[1] Despite their popularity, fully understanding the impact these adaptive learning rates have on convergence remains an area of ongoing research [21, 25, 27]. Notably, coordinate-wise learning rates make these methods sensitive to orthonormal reparameterization, distinguishing them from equivariant methods like gradient descent, whose performance does not change under orthonormal reparameterization.

Orthonormal reparameterizations correspond to full-dimensional changes of basis, and include seemingly benign transformations such as rotations of the loss landscape about the origin. The sensitivity of adaptive algorithms to rotation means that changes of basis can affect the rate of convergence to local minima, and even impact the generalization properties of the obtained solutions in the presence of non-convex landscapes. Given the sensitivity of these ubiquitous optimization methods to choice of basis, we pose the research question:

When using an adaptive algorithm, in what settings and to what extent can change-of-basis improve optimization?

We address this question by identifying geometric properties of loss functions that govern

[1] We focus on adaptive algorithms with diagonal preconditioners, corresponding to the standard implementation of Adagrad, Adam, and variants in popular software packages. Full-matrix Adagrad, proposed by Duchi et al. [14], is equivariant but computes a matrix root on every iteration and is thus rarely used in practice.

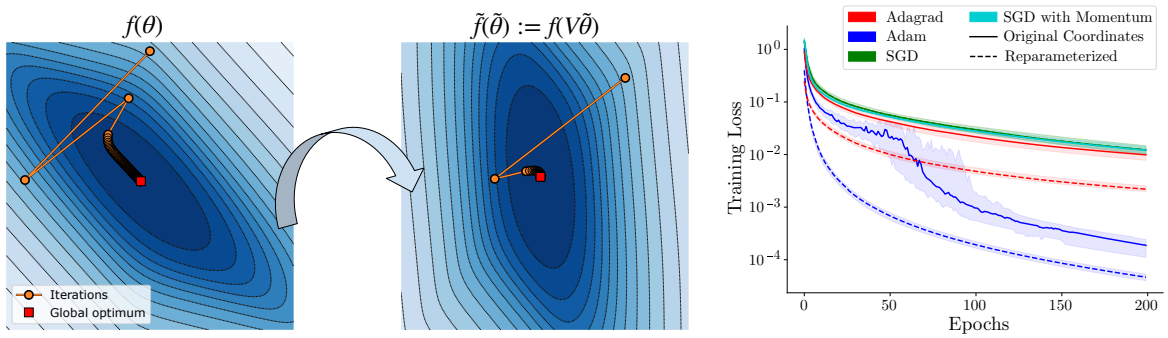


Figure 1: (Left) Using Adagrad in original coordinates and under EGOP reprameterization to optimize a two-dimensional log-sum-exp objective (defined in Section 6). In the EGOP eigenbasis, the primary directions of function variation are axis-aligned. Details in Section C.4. (Right) Cross-entropy loss from a 2-layer ReLU network in 2.4k dimensions trained on image classification using Adam, Adagrad, SGD, and SGD with momentum, in both original coordinates and under reparameterization. Equivariant methods (e.g. SGD) exhibit no change under reparameterization. Discussion and details in Section 6.

the sensitivity of adaptive algorithms to change-of-basis. We propose a reparameterization procedure based the *expected gradient outer product* (EGOP) matrix and show this reparameterization can improve convergence of adaptive methods. The geometric properties identified in this work—namely, strong decay of the EGOP eigenvalues—have been observed in a variety of machine learning objectives [29, 34, 42]. We include both empirical evidence and theoretical arguments suggesting that these properties arise when using natural data.

Contributions. We show that for a large class of objectives, the proposed reparameterization procedure can improve the convergence of adaptive optimization algorithms. [AD : Added new sentence:] Our analysis of EGOP spectral decay and low-rank structure yields a novel and specific hypothesis for why adaptive algorithms are particularly sensitive to changes of basis in machine learning settings. Our main contributions are as follows: **(1)** We characterize a class of objective functions for which reparameterization can reduce the number of iterations required for adaptive algorithms to converge to minima and first-order stationary points. **(2)** For these functions, we identify a choice of basis in which adaptive algorithms will perform well, and we propose an approximation procedure that only requires access to a (stochastic) gradient oracle, rather than analytical knowledge about the loss function. This procedure is defined in Section 2. **(3)** We develop theory that proves that in neighborhoods of local minima, the proposed reparameterization endows adaptive algorithms with improved convergence guarantees, quantified in terms of the spectrum of the EGOP matrix. Our main results are discussed in Section 3. **(4)** We empirically investigate this procedure and find that the proposed reparameterization improves the convergence of adaptive algorithms on a diverse suite of optimization problems. These experiments include results with large-scale contemporary neural network architectures. We present empirical results in Section 6.

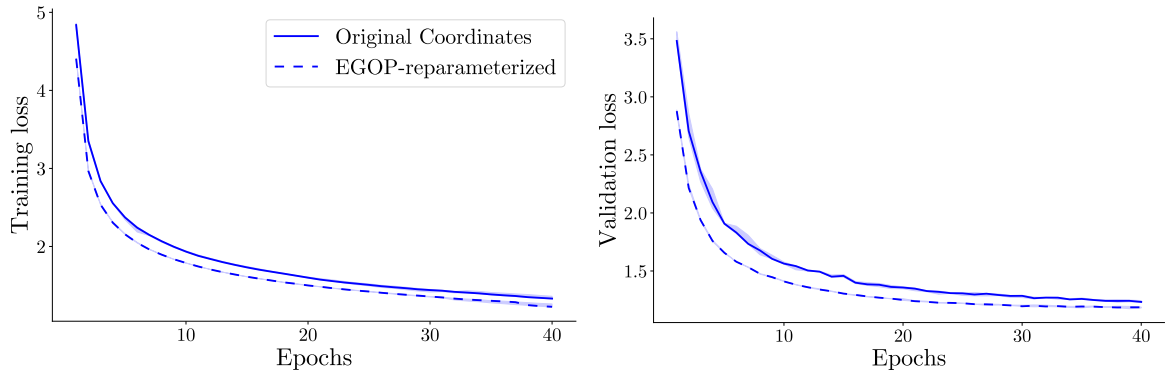


Figure 2: [AD : New figure.] EGOP reparameterization can scale to large, modern machine learning settings. Training and validation loss over epochs for a ResNet architecture trained to perform image classification on ImageNet. We compare optimizing the model with AdamW in original coordinates against AdamW under EGOP reparameterization. EGOP reparameterization improves convergence of both training and validation loss on this large-scale task. Results aggregated over five independent trials. Traces indicate medians, shaded regions indicate 25^{th}-75^{th} percentiles. Discussion and details in Section 6.

1.1. Related Work. Our work intersects with research on guarantees for adaptive algorithms, the role of orthogonal rotations in algorithmic performance, and geometry of machine learning objectives with natural data. Here we concisely survey related works, and we include an expanded discussion in Section F.

Geometric Sensitivity of Adaptive Methods. Recently, there has been renewed interest in distinguishing the properties of adaptive algorithms versus stochastic gradient descent (SGD), arising in part from several empirical studies suggesting that adaptive methods outperform SGD when training transformer models [23, 42]. Traditional analyses of adaptive algorithms establish regret bounds for online convex optimization [14, 19], and more recent work establishes convergence rates for smooth, non-convex objectives [11, 38]. However, because SGD is known to have optimal convergence rates in these settings, these theoretical results only show that adaptive algorithms achieve rates matching those of SGD.

In order to understand when adaptive algorithms enjoy provably stronger guarantees than SGD, recent work studies convergence under refined geometric assumptions, with particular emphasis on assumptions that are *not* rotationally invariant [21, 25, 40]. Xie et al. [40] establish convergence guarantees in terms of the ℓ_∞ smoothness constant and show experimentally that rotationally invariant geometric assumptions do not suffice to capture settings when Adam outperforms SGD. Jiang et al. [21] and Liu et al. [25] study convergence of Adagrad on objectives that are *coordinate-wise smooth*, defined in Section 1.2. Our analysis builds on these results; Jiang et al. [21] and Liu et al. [25] show that the sum of the coordinate-wise smoothness constants governs Adagrad convergence, and we prove that EGOP reparameterization decreases this value by a factor as large as $1/d$.

Change-of-Basis for Adaptive Algorithms. Recent works propose that using different orthonormal transformations with Adam and its variants can reduce computational costs and improve performance in neural networks [27, 37, 43]. Vyas et al. [37] proposed a method called SOAP, which computes an orthonormal reparameterization based on the singular vectors of the matrix-valued network gradients and performs optimization in this basis. Vyas et al. [37] empirically examine the performance of SOAP and find that it outperforms both Adam and Shampoo in LLM pre-training. Zhao et al. [43] propose GaLore, a method that simultaneously performs reparameterization and dimensionality reduction. GaLore computes a similar orthogonal basis to that used in SOAP, but instead of a full-dimensional change-of-basis GaLore retains only leading basis vectors in order to reduce dimension [43]. Maes et al. [27] empirically study Adam's rotational sensitivity and propose an orthonormal reparameterization, similar to those used by SOAP and GaLore; they show empirically that this can improve Adam's performance [27].

[AD : New paragraph:] EGOP reparameterization is related to SOAP, Shampoo, and GaLore because all three methods use spectral information of some notion of gradient covariance, but the properties of the EGOP reparameterization proposed in this work are fundamentally distinct from those of Shapmoo/SOAP/GaLore. As we discuss in Section 6, the change of basis leveraged by these methods is constrained to have a Kronecker product structure, while our approach does not impose this constraint; in Figure 8, we demonstrate empirically that as a result, EGOP reparameterization outperforms Shampoo and SOAP in settings where the Hessian is not well-approximated by a Kronecker product. Furthermore, as shown in Figure 9, we observe empirically that EGOP reparameterization can outperform SOAP and Shampoo even in the presence of exact Kronecker product structure, suggesting that EGOP spectral decay is a powerful tool even when additional structure is present.

Outside of training neural networks, several works have considered data-driven dimensionality reduction methods for optimizing more general objectives with low-rank EGOP matrices [3, 7]. These procedures target objectives with exact low-rank structure, while our method can improve convergence of adaptive algorithms even when the EGOP matrix has strong spectral decay but is full-rank.

EGOP Structure in Machine Learning. Increasing empirical evidence across a wide range of applications, including language modeling and image classification, suggests that empirical loss functions used for training machine learning models are approximately low-rank—meaning these functions vary strongly in only a small subset of directions in parameter space [29, 34, 42]. This approximate low-rank structure can be detected and analyzed using *active subspace methods*, which often leverage the *EGOP matrix*, defined in Section 2 [6]. Recently, there has been growing interest in the EGOP matrix in machine learning research [9, 28, 31, 44]. Radhakrishnan et al. [31] provide theoretical and empirical evidence that the weights of neural networks trained with gradient descent correlate strongly with a kind of EGOP matrix.[2]

1.2. Notation.

For a vector $\theta \in \mathbb{R}^d$, we denote its i^{th} entry by $\theta(i)$. We denote the inner product on $\mathbb{R}^d$ by $\langle \cdot, \cdot \rangle$, and let $\| \cdot \|_p$ denote the vector p-norm on $\mathbb{R}^d$, with $\|\theta\|_\infty \stackrel{\text{def}}{=} \max_i |\theta(i)|$.

[2] In Mallinar et al. [28] Radhakrishnan et al. [31], and Zhu et al. [44], the EGOP is defined using the gradient with respect to the *input data* instead of the optimization parameters.

Given a matrix $A \in \mathbb{R}^{m\times n}$, we write $\|A\|_{\mathsf{F}}$ for its *Frobenius* norm and $\|A\|_{\text{op}} \stackrel{\text{def}}{=} \sup_{\|\theta\|_2=1} \|A\theta\|_2$ for its *operator* norm. Given a PSD matrix $H \in \mathbb{R}^{d\times d}$, we denote the norm $\|\theta\|_H \stackrel{\text{def}}{=} \sqrt{\langle\theta, H\theta\rangle}$. For a matrix $A \in \mathbb{R}^{n\times m}$, we denote the vectorization of A by $\text{vec}(A) \in \mathbb{R}^{nm}$.

We obtain guarantees in terms of the *coordinate-wise smoothness constants* of the objective $f(\cdot)$. Following Jiang et al. [21] and Liu et al. [25], we say that a function f is *coordinate-wise smooth* within a set $\Theta \subseteq \mathbb{R}^d$ with respect to constants $L_1, \ldots, L_d > 0$ if $\forall\theta_1, \theta_2 \in \Theta$:

$$|f(\theta_1) - f(\theta_2) - \langle\nabla f(\theta_2), \theta_1 - \theta_2\rangle| \leq \frac{1}{2}\|\theta_1 - \theta_2\|_L^2, \tag{1.1}$$

where $L = \text{diag}(L_1, \ldots, L_d)$.

2. EGOP Reparameterization. Given a function $f : \mathbb{R}^d \to \mathbb{R}$ and a sampling distribution ρ, the expected gradient outer product of $f(\cdot)$ with respect to ρ is defined as

$$\text{EGOP}(f) \stackrel{\text{def}}{=} \mathbb{E}_{\theta\sim\rho}\left[\nabla f(\theta)\nabla f(\theta)^{\mathsf{T}}\right]. \tag{2.1}$$

As $f : \mathbb{R}^d \to \mathbb{R}$, the EGOP is a $d\times d$ symmetric matrix, and it is positive semi-definite because it is an expectation over PSD matrices. We denote its eigendecomposition by $\text{EGOP}(f) = V\Lambda V^{\mathsf{T}}$ where $V \in \mathbb{R}^{d\times d}$ is an orthonormal matrix. The EGOP eigenbasis captures key geometric qualities of the function $f(\cdot)$. When the sampling distribution ρ is isotropic, the leading eigenvectors of the EGOP matrix capture the directions of greatest variation in $f(\cdot)$, whereas the eigenspaces of the smallest eigenvalues are directions along which $f(\cdot)$ does not vary strongly: this is reflected by the fact that for any eigenpair (λ_i, v_i) of $\text{EGOP}(f)$, $\lambda_i = \mathbb{E}_{\theta\sim\rho}[\langle\nabla f(\theta), v_i\rangle^2]$.

The EGOP matrix is defined with respect to a user-specified sampling distribution ρ. Our guarantees in Section 3 require that ρ be isotropic and that its scale is large enough with respect to the norm of some local minimum of $f(\cdot)$. In Section 6, we present empirical results when the EGOP is estimated with ρ a standard Gaussian, and when ρ is defined by common neural network initialization distributions [15].

In this work, we compare how adaptive optimization algorithms perform when optimizing $f(\cdot)$ versus the reparameterized function $\tilde{f} : \mathbb{R}^d \to \mathbb{R}$ defined $\tilde{f}(\tilde{\theta}) \stackrel{\text{def}}{=} f(V\tilde{\theta})$. For any f, the objective $\tilde{f}(\cdot)$ can be approximated by empirically estimating the EGOP via Monte Carlo sampling and forming the eigenvectors of the empirical EGOP matrix, as summarized in Algorithm 2.1. Note that as V is orthogonal, any solution $\tilde{\theta}$ obtained by optimizing $\tilde{f}(\cdot)$ can be transformed into a solution in original coordinates as $\theta \stackrel{\text{def}}{=} V\tilde{\theta}$, which satisfies $f(\theta) = \tilde{f}(\tilde{\theta})$.

Algorithm 2.1 Reparameterization by EGOP Eigenbasis

1: **input:** M number of gradient samples, distribution ρ.
2: Generate $\{\theta_i\}_{i=1}^M \sim \rho$ i.i.d.
3: Form empirical EGOP $\hat{P} = \frac{1}{M}\sum_{i=1}^M \nabla f(\theta_i)\nabla f(\theta_i)^{\mathsf{T}}$
4: Form eigendecomposition $V\Lambda V^{\mathsf{T}} = \hat{P}$
5: Define function $\tilde{f}(\cdot) = f \circ V$
6: Optimize $\tilde{f}(\cdot)$ with adaptive algorithm of choice.
7: **return** $\tilde{\theta} \in \mathbb{R}^d$ the result of optimizing $\tilde{f}(\cdot)$.

3. Convergence Guarantees under EGOP Reparameterization. In this section, we show that for objectives with strong EGOP spectral decay and Lipschitz Hessians, there is some neighborhood around each local minima such that EGOP reparameterization improves Adagrad's convergence guarantees within that neighborhood. Our results show that the radius of this neighborhood scales inversely with the Lipschitz constant of the Hessian. We analyze Adagrad's convergence in both convex and nonconvex settings, focusing on the case of exact gradients for simplicity (see Algorithm A.1 for a precise statement). In the convex setting, we study constrained Adagrad, which uses the following update rule:

$$\texttt{(Adagrad)} \qquad \theta_{t+1} = \Pi_{\Theta}^{H_t}(\theta_t - \eta H_t^{-1}\nabla f(\theta_t)), \quad H_t := \operatorname{diag}\left(\epsilon^2 I_d + \sum_{j\le t} \nabla f(\theta_j)\nabla f(\theta_j)^{\mathsf{T}}\right)^{1/2},$$

where Π_{Θ}^{H} denotes the orthogonal projection onto set Θ under the metric induced by $H \succ 0$.

Our results relate the improvement obtained through reparameterization by the EGOP eigenbasis to the *stable rank*[3] of f:

$$\operatorname{sr}(f) \stackrel{\text{def}}{=} \frac{\sum_{i=1}^{d}\sqrt{\lambda_i(\mathrm{EGOP}(f))}}{\sqrt{\lambda_{\max}(\mathrm{EGOP}(f))}}. \tag{3.1}$$

Functions with strong EGOP spectral decay will have constant stable rank (tending towards 1 as spectral decay increases), while functions without EGOP spectral decay will have stable rank scaling with d.

We now introduce the main assumptions for our theoretical results. Setting the stage, we consider a twice-differentiable objective $f : \mathbb{R}^d \to \mathbb{R}$, a sampling distribution $\rho(\cdot)$ and fix a local minimum θ^*. We also consider a set $\Theta \subseteq \mathbb{R}^d$, and denote the ℓ_p diameter of Θ by $D_p \stackrel{\text{def}}{=} \max_{\theta,\theta'\in\Theta}\|\theta - \theta'\|_p$.

Assumption 1. *The set Θ is convex, and the sampling distribution $\rho(\cdot)$ has support contained in Θ and is an isotropic distribution centered at some $\theta_c \in \Theta$ with scale c. This implies $\mathbb{E}_\rho[\theta] = \theta_c$ and*

$$\mathbb{E}_\rho[(\theta - \theta_c)(\theta - \theta_c)^T] = c^2\mathbb{I}.$$

Moreover we assume Θ contains θ^* some local minimum of $f(\cdot)$ such that $\|\theta_c - \theta^*\|_2 \le c$.

Following the definitions introduced in Section 2, we let V denote the eigenbasis of the EGOP matrix and let $\tilde{f} \stackrel{\text{def}}{=} f \circ V$. Let $\tilde{\Theta}$ denote the corresponding transformation of set Θ: $\tilde{\Theta} \stackrel{\text{def}}{=} \{V^{\mathsf{T}}\theta \mid \theta \in \Theta\}$. Let $\{L_i\}_{i=1}^d$ denote the values for which $f(\cdot)$ is coordinate-wise smooth within Θ, following the definition in Eq. 1.1, and let $\{\tilde{L}_i\}_{i=1}^d$ denote the analogous coordinate-wise smoothness constants of $\tilde{f}(\cdot)$ within $\tilde{\Theta}$. We denote the vectors of coordinate-wise smoothness constants in original and reparameterized coordinates respectively by $\vec{L} \in \mathbb{R}^d$ and $\vec{\tilde{L}} \in \mathbb{R}^d$.

[3] We call this quantity the stable rank because of the connection to the stable rank considered in numerical linear algebra; the ratio in (3.1) is related to $\|G\|_*/\|G\|_{\mathrm{op}}$, where G denotes the empirical gradient bundle matrix $[\nabla f(\theta_1), \ldots, \nabla f(\theta_M)] \in \mathbb{R}^{d\times M}$ and $\|\cdot\|_*$ denotes the nuclear norm. This ratio is often referred to in literature as the *stable* or *effective rank* of G [5, 33].

To measure the density of the eigenvectors of EGOP(f), we introduce the values β_k: for v_k the k-th eigenvector of EGOP(f), let $\beta_k \stackrel{\text{def}}{=} \|v_k\|_1^2/d$. Note that $\forall k \in [d]$, $\beta_k \in [1/d, 1]$ with $\beta_k = 1/d$ if v_k is one-hot, and $\beta_k = 1$ if v_k is uniformly dense.

Our second assumption requires that the Hessian of $f(\cdot)$ be Lipschitz within Θ and that its Lipschitz constant be suitably bounded.

Assumption 2. *The Hessian of $f(\cdot)$ is H-Lipschitz within Θ: $\|\nabla^2 f(\theta_1) - \nabla^2 f(\theta_2)\|_{\text{op}} \leq H\|\theta_1 - \theta_2\|_2$ $\forall \theta_1, \theta_2 \in \Theta$, and $\exists \delta \in [0, \beta_1^2)$ such that H satisfies*

$$H \leq \frac{\sqrt{\delta\lambda_1(\text{EGOP}) + D_2^2\lambda_1^2(\nabla^2 f(\theta^*))} - D_2\lambda_1(\nabla^2 f(\theta^*))}{D_2^2} \tag{3.2}$$

[AD : New paragraph:] Assumption 2 quantifies how small the Lipschitz constant of the Hessian must be in order for the results in Theorem 3.1 to apply, and its particular form derives from the analysis. As long as the function has similar curvature within the constraint set Θ (as quantified by Eq. 3.2), the proposed reparameterization is guaranteed to lead to an improvement. Critically, H is not an input to the algorithm and Adagrad in the new coordinate system is guaranteed to converge even when Eq. 3.2 does not hold, albeit at a rate commensurate with that achieved in the original coordinate system.

Under the above assumptions, we show that EGOP spectral decay governs the improvements conferred by EGOP reparameterization. We first consider the convex setting. For simplicity, we instantiate these guarantees for constraint set Θ a ball; in practice, adaptive algorithms are often deployed with weight decay, which bounds the norm of the optimization parameters and effectively constrains θ to some ball. In Section A.1.1, we state guarantees for generic convex constraint sets.

Theorem 3.1 (Convergence for convex objectives with noise-free gradients). *Consider $f(\cdot)$ a convex objective, constraint set Θ a ball, and sampling distribution $\rho(\cdot)$ satisfying Assumptions 1 and 2, and $\tilde{f}(\cdot)$, $\tilde{\Theta}$ the EGOP-reparameterized objective and constraint set, respectively. Running Algorithm A.1 with constrained updates and inputs $(\tilde{f}(\cdot), \nabla\tilde{f}(\cdot), \tilde{\theta}_0, T, \epsilon, \tilde{\Theta})$ for any initial condition $\tilde{\theta}_0 \in \tilde{\Theta}$ produces iterates $\{\tilde{\theta}_t\}_{t=1}^T$ satisfying*

$$\frac{1}{T}\sum_{t=0}^{T-1}(\tilde{f}(\tilde{\theta}_t) - \tilde{f}(\tilde{\theta}^*)) = O\left(\left(\eta + \frac{D_2^2}{\eta}\right)^2 \frac{\|\vec{L}\|_1}{T}\left(\frac{\text{sr}(f)\sqrt{1+\delta}}{d(\beta_1 - \delta)} + \frac{\sqrt{\delta(1+\delta)}}{\beta_1 - \delta}\right) + \frac{\epsilon D_2^2}{\eta T}\right). \tag{3.3}$$

We compare the above bound with the convergence guarantee for Adagrad in original coordinates initialized at any $\theta_0 \in \Theta$:

$$\frac{1}{T}\sum_{t=0}^{T-1}(f(\theta_t) - f(\theta^*)) \leq \left(\eta + \frac{D_2^2}{\eta}\right)^2 \cdot \frac{\|\vec{L}\|_1}{T} + \frac{\epsilon D_2^2}{\eta T}. \tag{3.4}$$

When $f(\cdot)$ has strong EGOP spectral decay and dense leading eigenvectors, Theorem 3.1 implies reparameterization can significantly improve Adagrad's convergence bounds. In these settings, $\text{sr}(f)/(\beta_1 d) = O(1/d)$, implying stronger guarantees over original coordinates by up to a factor of d; the term $\|\vec{L}\|_1$ appearing in both Eq. 3.3 and 3.4 is the same, and denotes

the sum of the coordinate-wise smoothness constants of $f(\cdot)$ in original coordinates. Proof deferred to Section A.

In Section 4, we show that the two properties of the EGOP emphasized in this result, namely low stable rank and dense leading eigenvectors, are satisfied empirically for benchmark objectives. The factor $\beta = \|v_1\|_1^2/d$ tends towards 1 as the leading EGOP eigenvector gets denser. For the guarantee to reflect a benefit from reparameterization, it suffices to have $\beta \gg 1/d$ and small stable rank. We note that the density condition $\beta \gg 1/d$ is satisfied with high probability for random unit vectors in high dimensions: in particular, for $v \sim \text{Unif}(S^{d-1})$ where S^{d-1} is the unit sphere in $\mathbb{R}^d$, with high probability $\|v\|_1^2/d > 0.6$. For simplicity, the above guarantee only considers the density of v_1, but generalized guarantees in terms of β_k can be obtained. Similarly, one generalize Theorem 3.1 for approximate versions of the EGOP eigenbasis; such guarantees will scale with the subspace distance between the EGOP eigenbasis and its approximation. We note that numerical stability parameter ϵ is typically chosen small enough that the second term, $\epsilon D_2^2/(\eta T)$, is not the dominant term.

Many naturally-motivated objectives in machine learning have locally Lipschitz Hessians, including loss functions used in logistic regression, over-parameterized matrix factorization, and training of multilayer linear networks; see Section E.1.1 for examples.

In the non-convex setting, we study the convergence of unconstrained Adagrad (Algorithm A.1 with unconstrained updates). We show there is some neighborhood around each local minima such that while iterates remain in this neighborhood, EGOP-reparameterized Adagrad enjoys local convergence bounds that are stronger than those in original coordinates by a factor of $\text{sr}(f)/d\beta_1$ (see Theorem A.9). Similarly to the convex setting, the radius of these neighborhoods grows as the Lipschitz constant of the Hessian decreases.

4. EGOP Spectral Decay in Machine Learning.

Our analysis in Section 3 shows EGOP spectral decay and dense leading EGOP eigenvectors are sufficient for EGOP reparameterization to improve convergence guarantees for Adagrad. Here we present empirical evidence that these conditions occur in benchmark machine learning objectives and discuss why natural data may produce EGOP spectral decay in real-world problems.

Figure 3 shows the EGOP eigenspectrum for the objective function $f(\cdot)$ from an image classification problem. We use a 2-layer ReLU neural network to predict 10-class probabilities for handwritten digit images from the UCI ML digits dataset ([1]). Here $f(\cdot)$ denotes the negative log-likelihood loss on the training data. Figure 3 shows strong EGOP spectral decay. We plot λ_k/λ_1 for all EGOP eigenvalues λ_k on a logarithmic scale, while the inset zooms in on the leading 100 indices on a linear scale. These plots illustrate roughly exponential eigenvalue decay. In Section B.1, we show these trends are robust to the choice of sampling distribution (see Figure 10), that spectral decay persists when using *block* EGOP matrices, defined below in Section 5, and that decay occurs in other benchmark datasets. In Section B.1 we also examine the density of the leading EGOP eigenvectors for a 2-layer ReLU network on the UCI digits dataset and show that $\beta_k \gg 1/d$ for the leading eigenvectors (see Figure 12).

4.1. Natural Data Induces EGOP Spectral Decay.

In addition to empirical evidence, simple gradient calculations suggest natural data may induce EGOP spectral decay in machine learning problems. Many common objectives can be expressed as $f(\theta) = h(A\theta)$, where $h(\cdot)$ is a loss function and $A \in \mathbb{R}^{n\times d}$ is a data matrix whose rows comprise samples $a_i \in \mathbb{R}^d$. By the

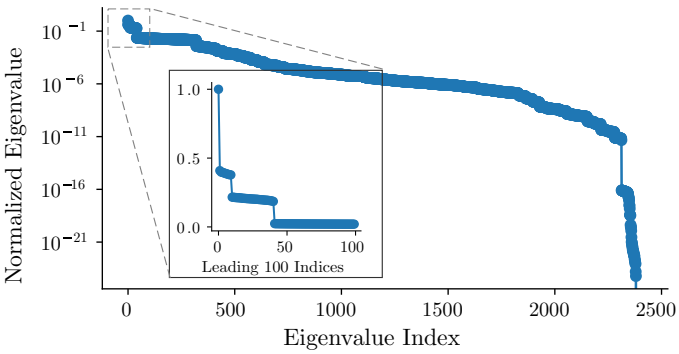


Figure 3: The EGOP eigenspectrum of a 2-layer ReLU network on the UCI handwritten digits dataset. Plot shows ratio λ_k/λ_1 as a function of eigenvalue index k, indexed in decreasing order. The EGOP eigenvalues decay sharply. In Figure 10, we demonstrate that these results are robust to the choice of sampling distribution ρ.

chain rule, the EGOP for such objectives satisfies

$$\text{EGOP}(f) = A^\mathsf{T} \mathbb{E}_{\theta\sim\rho}\left[\nabla_\theta h(A\theta)\nabla_\theta h(A\theta)^\mathsf{T}\right] A \tag{4.1}$$

where $\nabla_\theta h(A\theta)$ denotes the gradient $\nabla h(\cdot)$ evaluated at $A\theta$. This expression shows the EGOP is the transformation of some PSD matrix M by $A^\mathsf{T} M A$. It suggests that strong spectral decay in A may induce eigenvalue decay in the EGOP matrix. For many naturally occurring data distributions, the singular values of A exhibit strong decay [35]. The inner PSD matrix in the right hand side of Eq. 4.1 depends on both A and $h(\cdot)$, so without further assumptions on $h(\cdot)$ it is difficult to precisely characterize the spectral decay induced by the composition with A^T and A, but this can be quantified for specific choices of $h(\cdot)$; see Section A.2 for examples.

5. Heuristics for Scalability. Reparameterization with the EGOP eigenbasis incurs three main sources of additional computation: (1) sampling gradients to estimate the EGOP, (2) forming the EGOP eigenbasis, and (3) storing and applying the change-of-basis matrix to compute values and gradients of $f \circ V$. We outline some implementation details and heuristics that reduce the computational cost and enhance the scalability of the proposed framework. For a more detailed discussion, see Section D.

[AD : New paragraph] Empirically, we find that for functions with strong spectral decay, it suffices to accurately estimate only the leading EGOP eigenvectors. Based on this finding, one can use techniques like randomized SVD [18] to form $V_r \in \mathbb{R}^{d\times r}$, a matrix whose columns contain the estimated r leading eigenvectors of the EGOP. To enable optimization over the full parameter space, one can append a random orthogonal complementary basis to V_r, or introduce auxiliary variables restricted to the span of $V_r^\perp$. We refer to the latter approach as *auxiliary variable* EGOP reparameterization. We report empirical results using this heuristic in Figure 7, and we present full details and pseudocode in Section D.1.1.

For functions with strong spectral decay, a conservative number of gradient samples ($M \leq d$, where M is the number of samples and d is the number of problem parameters) is empirically

sufficient to yield change-of-basis matrices. In the presence of large EGOP spectral gaps, existing results formally establish that the number of gradient samples required to estimate the leading eigenspace grows logarithmically with dimension; see e.g., Corollary 3.8 in [6].

Structured or factorized approximations of V can also reduce the cost of storing and applying a change of basis. One can partition the variables in θ into subsets $\{S_i\}_{i=1}^L$ and perform *block reparameterization*. For each subset, we obtain a separate change-of-basis matrix $V^{(i)} \in \mathbb{R}^{|S_i| \times |S_i|}$ via the eigenvectors of the *block* EGOP matrix,

$$\text{EGOP}^{(i)} \overset{\text{def}}{=} \frac{1}{M} \sum_{k=1}^{M} \nabla_{S_i} f(\theta_k) \nabla_{S_i} f(\theta_k)^\mathsf{T}$$

where $\nabla_{S_i} f(\theta_k) \in \mathbb{R}^{|S_i|}$ is the vector of partial derivatives of f w.r.t. the entries in S_i, and the points $\{\theta_k\}_{k=1}^M$ are sampled i.i.d. from ρ. Block reparameterization is well-suited to multilayer neural networks, where each layer forms a parameter block; such choices are well-motivated by empirical observations about the block-diagonal structure of the Hessian in neural networks [13]. Cost can be further reduced by only reparameterizing a subset of blocks; see Figure 14c for results employing this heuristic.

[AD : New paragraph] When employing block-reparameterization, a related cost-reducing heuristic is to share reparameterization matrices V across different blocks of compatible dimensions; such a construction is well-motivated in convolutional layers, for example, where one can form blocks of parameters for each filter in a single layer. The key observation is that at initialization, the distribution of gradients from each filter is identical. Thus, the ground-truth EGOP matrix for each parameter block is identical, and so by estimating a single change-of-basis matrix V to be applied to all blocks in the convolutional layer, one can reduce both the number of forward/backward passes required to estimate the EGOP, as well as the storage cost of maintaining V [4]. Figure 2 demonstrates this heuristic efficiently scales to large architectures.

[AD : New paragraph] One natural extension of Algorithm 2.1 is to consider periodically re-estimating the EGOP, using a modified sampling distribution ρ which has been re-centered at the latest optimization iterate, θ_t. This allows one to recompute a change-of-basis matrix based on local geometry, and may be well-suited to problems with highly heterogeneous structure. In Figure 4, we report experimental results using periodic EGOP reparameterization for a large-scale, non-convex optimization problem and find that it empirically improves convergence. However, we do find that throughout our experiments, the up-front, single-time reparameterization proposed in Algorithm 2.1 is sufficient to improve convergence, even for objectives where the Hessian is *not* Lipschitz. See Section 6 for details.

6. Experimental Results. We examine the impact of EGOP reparameterization in a variety of settings. We compare how the seminal adaptive algorithms Adagrad and Adam perform when optimizing functions in original coordinates versus under reparameterization

[4] We note that while such a construction is equivalent to employing a Kronecker-product reparameterization matrix V, the effective Kronecker product imposed by this construction is different from that employed by SOAP/Shampoo. Moreover, the effective EGOP change-of-basis matrix applied to each convolutional filter does *not* have any structural constraint; see Section D.1.2 for details.

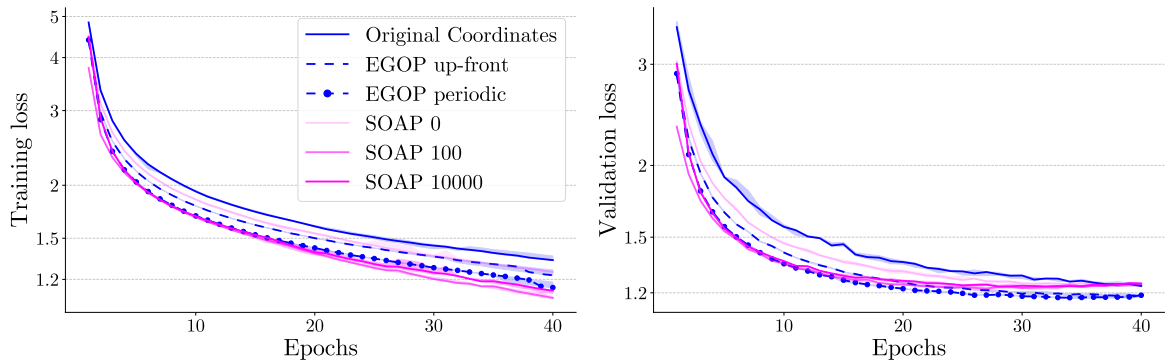


Figure 4: [AD : New figure] EGOP reparameterization can scale to large, modern machine learning settings. Training and validation loss over epochs for a ResNet architecture trained to perform image classification on ImageNet. We compare optimizing the model with Adam in original coordinates (blue, solid trace), under up-front EGOP reparameterization (blue, dashed trace), and with periodic EGOP reparameterization (blue, dashed trace with dots). We compare performance with SOAP under different update cadences; see discussion in the paragraph on *Large-scale Neural Networks* for descriptions of SOAP 0, SOAP 100, and SOAP 10000. EGOP reparameterization is competitive with SOAP at training on this large-scale task. Figure 5 demonstrates that EGOP reparameterization is competitive in wall-clock time as well. The results for Original Coordinates and EGOP up-front are identical to those in Figure 2, but in these plots we use logarithmic y-axis scaling for ease of comparing different traces.

by the empirical EGOP eigenbasis, both as described in Algorithm 2.1 and with the block reparameterization described in Section 5. We include comparisons to equivariant iterative optimization methods, namely (S)GD and (S)GD with momentum. When comparing methods, we always choose equivalent initializations: when a method in original coordinates is initialized at θ_0, chosen randomly, the reparameterized method is initialized at $V^\top \theta_0$. Full experimental details are in Section C.

Large-scale Neural Networks. [AD : New section (through line 354)] In Figure 4, we present results using EGOP reparameterization to improve convergence in large-scale residual networks (ResNets). We emulate the well-tuned architecture established in He et al. [20] and consider the task of image classification using the ImageNet dataset [12]. We consider training with AdamW, an adaptive algorithm and variant of Adam which implements weight decay decoupled from the momentum computation [26]. We compare AdamW in original coordinates with EGOP-reparameterized AdamW using a single up-front computation of the EGOP reparameterization matrix, as proposed in Algorithm 2.1. We find that up-front EGOP reparameterization improves both minimization of the training and validation loss compared to AdamW in original coordinates. We also consider results using EGOP reparameterization with periodic re-computation of the EGOP eigenbasis, as discussed in Section 3, where the EGOP eigenbasis is re-estimated every epoch. We find that periodic reparameterization offers

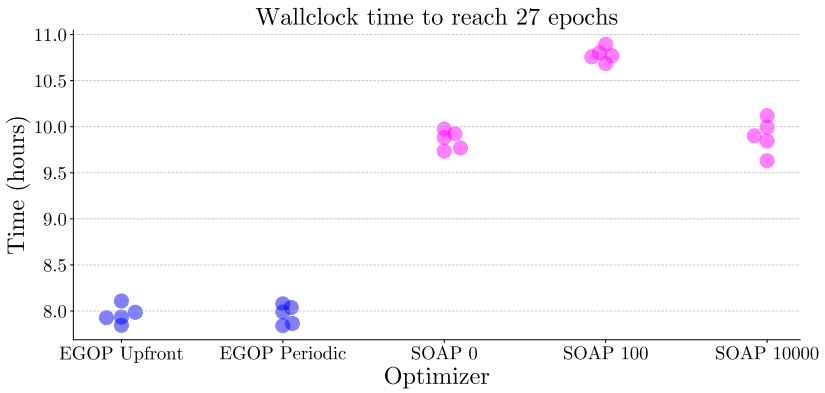


Figure 5: [AD : New figure] Wall-clock timing measures for training a residual network on the ImageNet dataset (counterpart to Figure 4). We report the wall-clock time required to train for 27 epochs, which is the number of epochs at which SOAP 100 minimizes its validation loss. As shown in Figure 4, at epoch 27 both EGOP methods achieve *lower* validation loss than SOAP. Both EGOP reparameterization methods achieve competitive wall-clock training times with SOAP and Shampoo. Horizontal jitter added for ease of visualization.

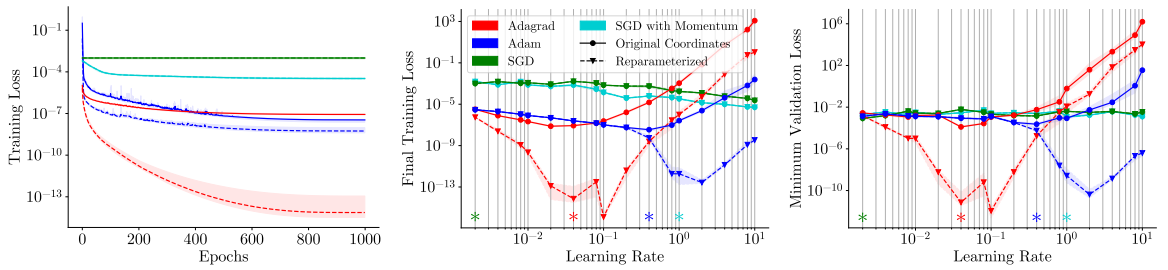
further benefits over up-front EGOP reparameterization in minimizing training loss, and that both variants achieve similar minimum validation loss. These results demonstrate the scalability and effectiveness of the heuristics discussed in Section 5, as well as demonstrating that EGOP reparameterization generalizes to a variety of modern neural network architectures.

We use this large-scale task to compare EGOP reparameterization with SOAP, which achieves state-of-the-art performance in training large neural networks [37]. We find that EGOP reparameterization is competitive both in performance and wall-clock time with this state-of-the-art method. In Figure 4, we present training and validation loss over epochs for both methods, and in Figure 5 we present wall-clock timing results. We find that EGOP-reparameterized AdamW achieves a lower minimum validation loss than SOAP, in addition to attaining competitive wall-clock time.

Our experiments include comparisons with SOAP at a variety of update cadences. In their original work proposing SOAP, Vyas et al. [37] reported that SOAP performance improves when the change-of-basis matrix is periodically updated, and they suggest an update cadence of every 10 to 100 batches. We include results with SOAP 0, which performs a single, up-front calculation of the change-of-basis matrix, comparable with up-front EGOP reparameterization; SOAP 100, which lies within the range of updates suggested by Vyas et al.; and SOAP 10k, which updates the change-of-basis matrix once per epoch, comparable with our implementation of periodic EGOP reparameterization.

Linear Feedforward Networks. We examine the impact of EGOP reparameterization on training 3-layer fully-connected linear feedforward networks using synthetic data. We consider parameters $\theta = [\mathrm{vec}(W_1), \mathrm{vec}(W_2), \mathrm{vec}(W_3)]$ and train by minimizing loss function

$$f(\theta) = \|W_3 W_2 W_1 A - Y\|_{\mathsf{F}}^2 / n_{\text{samples}} \tag{6.1}$$



(a) Training loss over epochs (b) Training loss over learning rates (c) Validation loss by learning rate

Figure 6: Training multilayer linear networks (6.1). Both SGD and SGD with momentum are equivariant optimization methods, so their results in original and reparameterized coordinates are exactly superimposed. In Figure 6c we consider the minimum validation loss achieved over epochs during training. Results are aggregated over 10 independent trials, with traces showing medians and shading indicating 25th-75th quartile. Asterisks indicate the learning rate used for each method in Figure 6a. Learning rates chosen to minimize validation loss of the algorithm in *original* coordinates.

where $A \in \mathbb{R}^{10 \times n_{\text{samples}}}$, and $Y = M^* A$ for $M^* \in \mathbb{R}^{10 \times 10}$ drawn from a standard Gaussian distribution. We use $W_1 \in \mathbb{R}^{50 \times 10}$, $W_2 \in \mathbb{R}^{30 \times 50}$, $W_3 \in \mathbb{R}^{10 \times 30}$. We induce spectral decay in EGOP(f) by generating A with singular values $\sigma_k(A) = k^{-2}$ and random singular vectors. We use minibatched stochastic gradient samples throughout. Full details are deferred to section C.

In Figure 6 we study the impact of global EGOP reparameterization (Alg. 2.1). Figure 6a shows training loss over epochs in original coordinates and under reparameterization. Reparameterized Adagrad and reparameterized Adam achieve significantly lower final training loss and faster convergence than their counterparts in original coordinates. The adaptive methods also outperform the equivariant methods (SGD and SGD with momentum) in both coordinate settings. Figure 6b demonstrates the robustness of these results across learning rates. Asterisks along the x-axis indicate the learning rates used in Figure 6a. Figure 6c confirms that the improved minimization of the training loss enabled by reparameterization does not lead to over-fitting; it shows that reparameterization enables adaptive methods to achieve better performance on validation data.

In Section B.2, we compare the above results to those obtained on the same task when employing several heuristics described in Section 5, including block reparameterization and reparameterizing only the first layer of the network. We find that both block-reparameterized Adagrad and Adam achieve lower final training loss than their counterparts in original coordinates. For Adagrad, reparameterizing only the first layer confers a benefit comparable to reparameterizing all layers, while for Adam reparameterizing the first layer alone provides only a marginal benefit over original coordinates.

ReLU Networks for Image Classification. We examine the impact of EGOP reparameterization when training networks to perform image classification using real-world data. We consider two benchmark image classification datasets: the UCI hand-written digits dataset

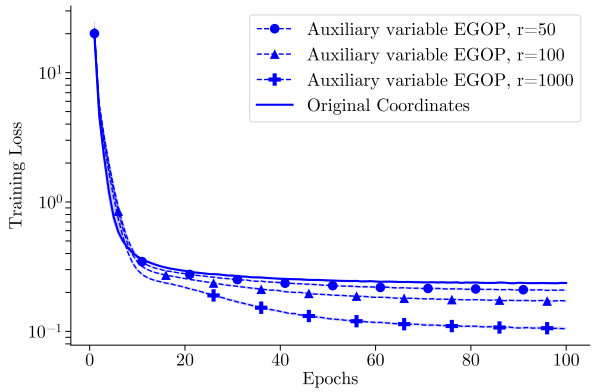


Figure 7: Scaling EGOP reparameterization to large fully-connected layers using auxiliary variable EGOP reparameterization, as described in Section 5. We perform reparameterization using varying numbers of leading EGOP eigenvectors, denoted by r. Even small values of r suffice to improve convergence, and the benefit increases as r increases. Results from training a ReLU network to classify the fashionMNIST dataset, in which the largest layer contains 78k weights. In Figure 16, we validate that this translates into improved wallclock time to convergence as well as improved accuracy on held-out data.

(8×8 pixel images), and the fashionMNIST dataset (28×28 pixel images of clothing) [1, 39]. We train multilayer fully-connected ReLU networks to perform image classification on each dataset by minimizing the cross-entropy loss.

We perform block EGOP reparameterization, as described in Section 5. For the UCI digits, the number of gradient samples drawn equals the number of model weight parameters. [AD : New text.] For the fashionMNIST results in Figure 7, we additionally employ heuristics described in Section 5. We perform auxiliary variable EGOP reparameterization, using randomized SVD to approximate the leading r EGOP eigenvectors, where $r \ll d$ the number of optimization variables. We use only a small number of stochastic minibatch gradients ($M = 0.01d$). These heuristics effectively scale EGOP reparameterization to large fully-connected layers. These heuristics also lead to faster convergence in terms of wall-clock time as reported in Figure 16.

Figure 1 (right) and Figure 7 plot training loss by epoch for the UCI digits and fashionMNIST respectively. On both datasets, reparameterized adaptive methods out-perform both their counterparts in original coordinates and the equivariant methods. In Section B.2 we report results on hold-out data, showing reparameterized methods generalize as well as or better than original coordinates, and show that these results are robust to the choice of learning rate. For full experimental details, see Section C.

Regression with Errors-in-Variables. [AD : New section (through lines 432)]In this section, we empirically demonstrate one distinction between our proposed reparameterization and related methods of SOAP and Shampoo; EGOP reparameterization exploits approximate low-rank loss function geometry, while SOAP and Shampoo target Kronecker-product structure in the Hessian.

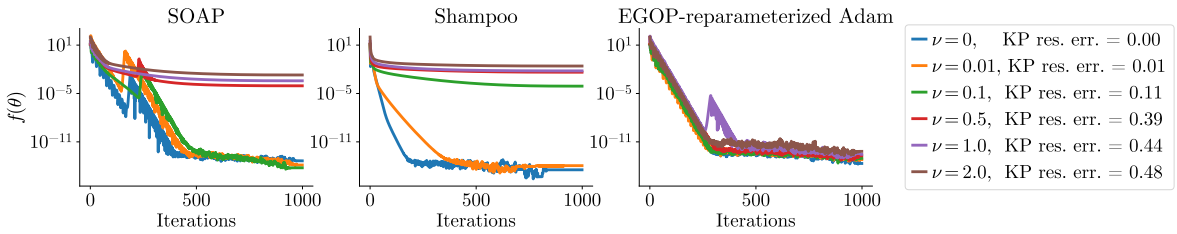


Figure 8: [AD : New figure] Comparing the impact of Hessian Kronecker-product (KP) structure on EGOP reparameterization, SOAP, and Shampoo, for objective $f(\theta)$ defined in (6.2). SOAP and Shampoo both exhibit worse convergence as the Hessian KP approximation worsens. We quantify Hessian KP structure via the KP residual error, defined in (6.3). EGOP-reparameterized Adam's convergence is robust to this structure.

Shampoo and SOAP are designed for optimization problems where the variable has a natural matrix structure (e.g., a neural network weight matrix), and thus the gradient is also matrix-valued. Their change-of-basis derives from the eigenvectors of the Gram matrices of these gradient matrices. When mapped back to a vector representation (i.e., the vectorization of the matrix parameter and the setting of our EGOP-based approach), the corresponding SOAP/Shampoo change-of-basis matrix is restricted to a Kronecker product form. As a result, Shampoo and SOAP are hypothesized to perform best when the Hessian admits a close Kronecker-product approximation. Figure 8 presents evidence supporting this hypothesis.

We consider an objective that allows smooth interpolation between regimes when the Hessian admits an exact Kronecker-product structure, and cases where the Hessian is poorly approximated by such a factorization. For parameters $\theta = \mathrm{vec}(W)$, $W \in \mathbb{R}^{n\times m}$, we consider

$$f(\theta) = \frac{1}{2}\|A(M \odot W) - Y\|_F^2. \tag{6.2}$$

Here, $M \in \mathbb{R}^{n\times m}$ is a fixed, known multiplicative noisy mask, $A \in \mathbb{R}^{n\times n}$ is the measurement map, and $Y = A(M \odot W^*)$ for some unknown $W^* \in \mathbb{R}^{n\times m}$. This models, for instance, regression problems based on data from sensors with well-characterized but potentially non-homogeneous gain.

Figure 8 displays $f(\cdot)$ by iteration for different instances of M. We draw the entries of M i.i.d. from $\mathcal{N}(1, \nu^2)$, where different problem instances use different ν. When $\nu = 0$, M is all ones, and the Hessian is exactly Kronecker-structured. As ν increases, the Kronecker structure of the Hessian degrades. We quantify this via Kronecker product residual error:

$$\text{KP residual}(f) = \min_{A\in\mathbb{R}^{n\times n}, B\in\mathbb{R}^{n\times n}} \|\nabla^2 f(\theta) - A \otimes B\|_F / \|\nabla^2 f(\theta)\|_F. \tag{6.3}$$

Figure 8 (left, center) shows that SOAP and Shampoo performance degrade as the Hessian becomes farther from Kronecker-product structured, while EGOP-reparameterized Adam Figure 8 (right) maintains linear convergence across all problem instances. These experiments allow SOAP and Shampoo to recompute their change-of-basis matrix on *every iteration*, whereas

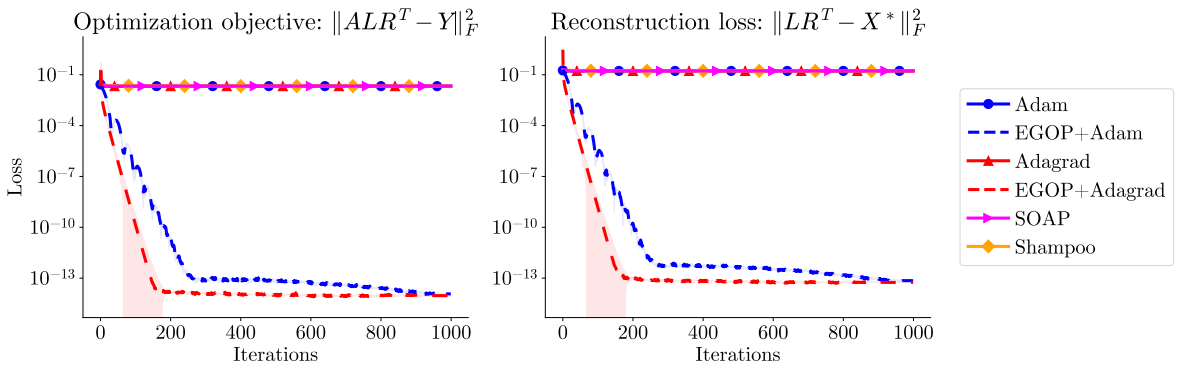


Figure 9: [AD : New figure] Comparing optimization algorithms for low-rank matrix factorization, as defined in (6.4). We note that base Adam, base Adagrad, Shampoo, and SOAP all have similar performance, and thus **their traces are superimposed**. On the right, we validate that the improved optimization exhibited by EGOP-reparameterized Adam and Adagrad selects a good minimizer, and does not overfit to noise. We emphasize that for this objective, the Hessian blocks $\nabla^2_{L,L} f(\theta)$ and $\nabla^2_{R,R} f(\theta)$ always admit exact Kronecker product factorizations, making this setting well-suited to the constraints imposed by SOAP and Shampoo.

for EGOP reparameterization we compute a single, up-front change-of-basis matrix following Algorithm 2.1. Even with this advantage, SOAP and Shampoo do not outperform EGOP-reparameterized Adam when the Hessian does not admit a good Kronecker approximation.

Low-rank matrix factorization. [AD : New section (through lines 451)] We complement the above results by showing EGOP reparameterization can offer advantages over SOAP and Shampoo even when the Hessian admits an exact Kronecker product factorization. We consider low-rank matrix factorization; for parameters $\theta = [\mathrm{vec}(L), \mathrm{vec}(R)]$ for $L \in \mathbb{R}^{n\times r}, R \in \mathbb{R}^{n\times r}$, we minimize the objective

$$f(\theta) = \frac{1}{2}\|A(LR^T) - Y\|_F^2 \tag{6.4}$$

where $Y = AX^* + \zeta$ is a matrix of observations with noise $\zeta_{i,j} \sim \mathcal{N}(0, 0.001)$, $X^* \in \mathbb{R}^{n\times n}$ is an unknown ground-truth matrix of (known) rank r, and $A \in \mathbb{R}^{n\times n}$ is a measurement map with singular value decay $\sigma_k(A) = k^{-1/2}$. We use standard spectral initialization.

In Figure 9, we plot the value of $f(\cdot)$ over iterations for varying algorithms, including SOAP and Shampoo, base Adam and Adagrad, and EGOP-reparameterized Adam and Adagrad. EGOP-reparameterized Adam and Adagrad both achieve better minimization of $f(\cdot)$, and also yield lower reconstruction error, as quantified by $\|LR^T - X^*\|_F^2$. As before, we allow both SOAP and Shampoo to update their change-of-basis at every iteration, whereas EGOP reparameterization performs only a single up-front computation of the change-of-basis matrix. For this objective, the Hessian blocks $\nabla^2_{L,L} f(\theta)$ and $\nabla^2_{R,R} f(\theta)$ always admit exact Kronecker product decompositions, making this an objective well-suited to the constraints imposed by SOAP and Shampoo. Nonetheless, EGOP-reparameterized algorithms outperform SOAP/Shampoo even in this favorable regime, indicating the need for further analysis

in order to predict when each approach will be optimal.

7. Conclusions and Limitations. In this work, we have shown through both analysis and experiments that EGOP reparameterization can improve the convergence of adaptive algorithms. There may be opportunities for future work to explore other reparameterizations and additional hyperparameter tuning, which we have not examined in this work. One limitation of this work is the engineering required for scalability, as discussed in Section 5. Fully characterizing the trade-off between improved convergence and the up-front cost of reparameterization remains a direction for future research. [AD : New text (through end of conclusion)] Another interesting direction for future work would be studying whether the guarantees in Theorem 3.1 can be strengthened under a periodic EGOP reparameterization scheme, as described in Section 5. Our experiments in Figure 4 demonstrate empirically that such periodic re-estimation of the EGOP eigenbasis can offer further improvements over the up-front reparameterization analyzed in this work.

Further work examining the tradeoffs between EGOP reparameterization and the reparameterizations employed by SOAP and Shampoo would benefit practitioners by providing greater guidance on when each method may perform best. Our empirical results in Figure 9 demonstrate that EGOP reparameterization can outperform SOAP and Shampoo even in the presence of exact Kronecker product structure, suggesting that approximate low-rank geometry, as quantified by EGOP spectral decay, is a powerful tool even when additional geometric structure is present.

Appendix A. Proofs. We briefly introduce additional notation used in our analysis. We denote the unit sphere in d dimensions by $\mathbb{S}^{d-1}$ and the Euclidean ball of radius r centered at $\bar{x}$ by $\mathcal{B}(\bar{x}; r)$. For matrices $A, B \in \mathbb{R}^{d\times d}$, we denote the Löwner ordering by $A \preceq B$. We write $A \lesssim B$ to indicate the existence of a dimension-independent positive constant $c > 0$ such that $A \leq cB$. We write $\mathcal{O}(d, r) := \{X \in \mathbb{R}^{d\times r} \mid X^\mathsf{T} X = I_r\}$ and $\mathcal{O}(d) \equiv \mathcal{O}(d, d)$ for the set of matrices with orthogonal columns. We write $\langle X, Y\rangle \stackrel{\text{def}}{=} \mathrm{tr}(X^\mathsf{T} Y)$ for the Euclidean inner product and $\|X\| = \sqrt{\langle X, X\rangle}$ for its induced norm.] We write $\mathcal{O}(d)$ for the set of matrices with orthogonal columns. Given a set of scalars $\{L_i\}_{i=1}^d$, we let $\mathrm{diag}(L_1, \ldots, L_d)$ denote the $d \times d$ diagonal matrix with values $L_1, \ldots, L_d$ along the diagonal. We let $\mathrm{mat}(\theta) \in \mathbb{R}^{m\times n}$ denote the reshaping of θ into a matrix for some m and n such that $mn = d$, and similarly we let $\mathrm{vec}(M)$ denote the reshaping of a matrix into a vector.

Following the definitions introduced in Section 2, we let V denote the eigenbasis of the EGOP matrix and let $\tilde{f} \stackrel{\text{def}}{=} f \circ V$. Let $\tilde{\Theta}$ denote the corresponding transformation of set Θ: $\tilde{\Theta} \stackrel{\text{def}}{=} \{V^\mathsf{T}\theta \mid \theta \in \Theta\}$. We denote the corresponding diameter measurements as $\tilde{D}_p \stackrel{\text{def}}{=} \max_{\tilde{\theta}_1, \tilde{\theta}_2 \in \tilde{\Theta}} \|\tilde{\theta}_1 - \tilde{\theta}_2\|_p$. Let $\{L_i\}_{i=1}^d$ denote the values for which $f(\cdot)$ is coordinate-wise smooth within Θ, following Eq. 1.1, and let $\{\tilde{L}_i\}_{i=1}^d$ denote the analogous coordinate-wise smoothness constants of $\tilde{f}(\cdot)$ within $\tilde{\Theta}$. We denote the vectors of coordinate-wise smoothness constants in original and reparameterized coordinates respectively by $\vec{L} \in \mathbb{R}^d$ and $\vec{\tilde{L}} \in \mathbb{R}^d$.

We analyze the convergence of Adagrad in both convex and nonconvex settings. For completeness, we recall the full Adagrad algorithm in Algorithm A.1.

Algorithm A.1 Adagrad$(f, g, \theta_0, T, \epsilon : \texttt{default} = 10^{-8}, \Theta : \texttt{Optional})$

Input: Objective f, oracle gradient g, $\theta_0 \in \mathbb{R}^d$, η step size, T number of iterations, ϵ numerical stability parameter, $\Theta \subseteq \mathbb{R}^d$ constraint set.
$v_0 \leftarrow \epsilon^2 \mathbb{I}_d$
for $t \in [1, T]$ **do**
  $g_t \leftarrow g(f, \theta_t)$
  $v_t(i) \leftarrow v_{t-1}(i) + g_t(i)^2 \ \forall i \in [d]$
  $H_t \leftarrow \mathrm{diag}(v_t)^{1/2}$
  $w_t \leftarrow \theta_t - \eta H_t^{-1} g_t$
  Constrained update: $\theta_{t+1} = \Pi_\Theta^{H_t}(w_t)$
  Unconstrained update: $\theta_{t+1} = w_t$
end for
Output: $\theta_T \in \mathbb{R}^d$

A.1. Proofs from Section 3. Several recent works have proved convergence guarantees for Adagrad in terms of $\|\vec{L}\|_1$, showing that Adagrad enjoys better performance guarantees in settings when $\|\vec{L}\|_1$ is small [21, 25, 40]. Our main results establish that when EGOP(f) has strong spectral decay and dense leading eigenvectors, then reparameterization can produce $\|\vec{\tilde{L}}\|_1$ significantly smaller than $\|\vec{L}\|_1$, implying better convergence bounds.

Theorem A.1. *If $\exists \delta \in [0, \beta_1^2)$ such that the Hessian of $f(\cdot)$ is H-Lipschitz within Θ for H*

satisfying

$$H \leq \frac{\sqrt{\delta\lambda_1(\text{EGOP}) + D_2^2\lambda_1^2(\nabla^2 f(\theta^*))} - D_2\lambda_1(\nabla^2 f(\theta^*))}{D_2^2} \tag{A.1}$$

Then within Θ *and* $\tilde{\Theta}$ *respectively, the smoothness constants of* $f(\cdot)$ *and* $\tilde{f}(\cdot)$ *satisfy*

$$\frac{\|\vec{\tilde{L}}\|_1}{\|\vec{L}\|_1} = O\left(\sqrt{1+\delta}\left(\frac{\text{sr}(f)}{d(\beta_1-\delta)} + \frac{\sqrt{\delta}}{\beta_1-\delta}\right)\right)$$

where $\text{sr}(f)$ *denotes the stable rank of* $f(\cdot)$.

Thm. A.1 implies that when the EGOP leading eigenvectors have constant density, i.e. when $\beta_1 >> 1/d$, then for small δ the ratio $\|\vec{\tilde{L}}\|_1/\|\vec{L}\|_1$ scales as $\text{sr}(f)/d\beta_1$. We note that the density condition $\beta \gg 1/d$ is satisfied with high probability for random unit vectors in high dimensions: in particular, for $v \sim \text{Unif}(S^{d-1})$ where S^{d-1} is the unit sphere in $\mathbb{R}^d$, with high probability $\|v\|_1^2/d \approx 2/\pi > 0.6$.

We prove Theorem A.1 by establishing the following pair of claims. The first lower bounds the sum of the coordinate-wise smoothness constants of $f(\cdot)$ in original coordinates:

Lemma A.2. *Consider* $f(\cdot)$, Θ, ρ *satisfying Assumptions 1 and 2 and consider the set of dense unit vectors* $\nu \in \{\pm d^{-1/2}\}^d$, *the collection of vectors whose entries all have magnitude* $|\nu(i)| = d^{1/2}$. *Then within* Θ, $f(\cdot)$ *is coordinate-wise smooth with respect to constants satisfying*

$$\|\vec{L}\|_1 \geq \frac{d}{2c} \cdot \max_{\nu\in\{\pm d^{-1/2}\}^d} \frac{\langle \nu, \text{EGOP}\,\nu\rangle - \gamma}{\sqrt{\lambda_{\max}(\text{EGOP}) + \gamma}}$$

where

$$\gamma \stackrel{\text{def}}{=} 2H\lambda_{\max}(\nabla^2 f(\theta^*))M_3 + H^2 M_4 \tag{A.2}$$

where H *is the Lipschitz constant of the Hessian of* $f(\cdot)$ *in Assumption 2 and* $M_p \stackrel{\text{def}}{=} \mathbb{E}_{\theta\sim\rho}[\|\theta - \theta^*\|_2^p]$.

If the EGOP has dense leading eigenvectors, then the term $\langle \nu, \text{EGOP}\,\nu\rangle$ is large. In particular, Lemma A.2 implies the following:

Corollary A.3. *For* $f(\cdot)$, Θ,ρ *satisfying Assumptions 1 and 2, the smoothness constants of* f *satisfy*

$$\|\vec{L}\|_1 \geq \frac{d}{2c}\frac{\beta_k\lambda_k(\text{EGOP}) - \gamma}{\sqrt{\lambda_{\max}(\text{EGOP}) + \gamma}}.$$

where $\beta_k \stackrel{\text{def}}{=} \|v_k\|_1^2/d$ *for* $\lambda_k(\text{EGOP}), v_k$ *the* k^{th} *eigenvalue and eigenvector of* $\text{EGOP}(f)$, *and* γ *defined in Eq. A.2.*

In contrast, we show that under reparameterization by the EGOP eigenbasis, the smoothness constants can be upper bounded by the following:

Lemma A.4. *Let V be the eigenbasis of the EGOP of $f(\cdot)$ with respect to ρ, and define $\tilde{f}(\tilde{\theta}) \stackrel{\text{def}}{=} f(V\theta)$. Let the function $f(\cdot)$ satisfy Assumptions 1 and 2. Then within $\tilde{\Theta}$, $\tilde{f}(\cdot)$ satisfies Eq. 1.1 with respect to smoothness coordinates whose sum is bounded by*

$$\|\vec{\tilde{L}}\|_1 \leq d\left(\frac{\sqrt{\gamma}}{c} + HD_2\right) + \frac{1}{c}\sum_{i=1}^{d}\sqrt{\lambda_i}$$

where γ is defined in Eq. A.2. and λ_i denotes the ith eigenvalue of the EGOP of f w.r.t. ρ.

Note that as Lipschitz constant of Hessian in Assumption 2 goes to zero, so does the value of γ in both Lemmas A.3 and A.4. Theorem A.1 follows from Lemmas A.3 and A.4.

Dense EGOP eigenvectors lead to large $||\tilde{\mathbf{L}}||_1$. We defer the full proof of Lemma A.3 to Section E.1. We note here the main intermediate results used to establish Lemma A.3. The first intermediate result gives an alternative characterization of the coordinate wise smoothness constants defined in Eq. 1.1.

Lemma A.5. *A twice-differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ satisfies Eq. 1.1 with respect to smoothness constants $L_1, \ldots, L_d$ within Θ if and only if $\forall \theta \in \Theta, \forall v \in \mathbb{R}^d$,*

$$|\langle v, \nabla^2 f(\theta) v\rangle| \leq \langle v, \mathrm{diag}(L) v\rangle$$

where $\mathrm{diag}(L) \in \mathbb{R}^{d\times d}$ is the diagonal matrix with diagonal entries L_i.

The second intermediate result relates the EGOP to the Hessian.

Lemma A.6. *Consider twice-differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ and sampling distribution ρ satisfying Assumptions 2, and 1 with respect to local minimum θ^*. Then the EGOP of $f(\cdot)$ with respect to ρ satisfies*

$$\mathbb{E}_{\theta\sim\rho}[\nabla f(\theta)\nabla f(\theta)^\mathsf{T}] = G(\theta^*) + E_f(\theta^*)$$

where $G(\theta^*)$ is a PSD matrix satisfying

$$c^2\nabla^2 f(\theta^*)\nabla^2 f(\theta^*)^\mathsf{T} \preceq G(\theta^*) \preceq 2c^2\nabla^2 f(\theta^*)\nabla^2 f(\theta^*)^\mathsf{T}$$

and the matrix $E_f(\theta^*)$ satisfies

$$|\langle v, E_f(\theta^*) v\rangle| \leq \left(2H\lambda_{\max}(\nabla^2 f(\theta^*))M_3 + H^2 M_4\right)\|v\|_2^2$$

where

$$M_p \stackrel{\text{def}}{=} \mathbb{E}_{\theta\sim\rho}[\|\theta - \theta^*\|_2^p].$$

We defer the proof of these intermediate results to Section E.1.

Reparameterization by EGOP eigenbasis produces small $||\tilde{\tilde{\mathbf{L}}}||_1$. W defer the full proof of Lemma A.4 to Section E. Here we present results establishing the equivariance of the EGOP matrix, which is the key step in proving Lemma A.4. Recall that Assumption 1 implies ρ is isotropic about $\theta_c \in \Theta$. It is convenient to express ρ as

$$\rho(\theta) = \rho_0(\theta - \theta_c)$$

for ρ_0 mean-zero and isotropic (about the origin). For reparameterized set

$$\tilde{\Theta} \stackrel{\text{def}}{=} \{V^\mathsf{T}\theta \mid \theta \in \Theta\}$$

and element $V^\mathsf{T}\theta_c \in \tilde{\Theta}$, we consider the reparameterized sampling distribution

$$\tilde{\rho}(\tilde{\theta}) \stackrel{\text{def}}{=} \rho(\tilde{\theta} - V^\mathsf{T}\theta_c). \tag{A.3}$$

The distribution $\tilde{\rho}$ is then isotropic about $V^\mathsf{T}\theta_c$. Moreover, because ρ_0 is isotropic,

$$\tilde{\rho}(\tilde{\theta}) = \rho_0(\tilde{\theta} - V^\mathsf{T}\theta_c) = \rho_0(V(\tilde{\theta} - V^\mathsf{T}\theta_c)) = \rho_0(V\tilde{\theta} - \theta_c) = \rho(V\tilde{\theta})$$

and similarly $\rho(\theta) = \tilde{\rho}(V^T\theta)$.

Lemma A.7. *Given $f(\cdot)$ whose EGOP with respect to isometric distribution ρ has eigenvalue decomposition*

$$\mathbb{E}_{\theta\sim\rho}[\nabla f(\theta)\nabla f(\theta)^\mathsf{T}] = V\Lambda V^\mathsf{T},$$

the EGOP of the reparameterized function $\tilde{f}(\theta) \stackrel{\text{def}}{=} f(V\theta)$ with respect to reparameterized sampling distribution $\tilde{\rho}$ (defined in Eq. A.3), we have

$$\mathbb{E}_{\theta\sim\tilde{\rho}}[\nabla \tilde{f}(\theta)\nabla \tilde{f}(\theta)^\mathsf{T}] = \Lambda$$

We defer the proof of Lemma A.7 to Section E.

Proof of Theorem A.1. Combining Lemmas A.3 and A.4 implies Theorem A.1:

Proof of Theorem A.1. By Lemma A.3,

$$\|\vec{L}\|_1 \geq \max_k \frac{d}{2c}\frac{\beta_k\lambda_k(\text{EGOP}(f)) - \gamma}{\sqrt{\lambda_1(\text{EGOP}(f)) + \gamma}} \geq \frac{d}{2c}\frac{\beta_1\lambda_1(\text{EGOP}(f)) - \gamma}{\sqrt{\lambda_1(\text{EGOP}(f)) + \gamma}}$$

for

$$\gamma \stackrel{\text{def}}{=} 2H\lambda_{\max}(\nabla^2 f(\theta^*))M_3 + H^2M_4.$$

Recall that

$$M_p \stackrel{\text{def}}{=} \mathbb{E}_{\theta\sim\rho}[\|\theta - \theta^*\|_2^p]$$

and thus, because we assume $\text{supp}(\rho) \subseteq \Theta$, $M_p \leq D_2^p$ where D_2 is the ℓ_2 diameter of Θ.

Additionally, Lemma A.4 states that

$$\|\vec{\tilde{L}}\|_1 \leq d\left(\frac{\sqrt{\gamma}}{c} + HD_2\right) + \frac{1}{c}\sum_{i=1}^{d}\sqrt{\lambda_i(\text{EGOP}(f))}$$

where γ is the same as defined above.

Combining these two results implies

$$\frac{\|\vec{\tilde{L}}\|_1}{\|\vec{L}\|_1} \leq \left(d\left(\frac{\sqrt{\gamma}}{c} + HD_2\right) + \frac{1}{c}\sum_{i=1}^{d}\sqrt{\lambda_i}\right)\left(\frac{d}{2c}\frac{\beta_1\lambda_1 - \gamma}{\sqrt{\lambda_1 + \gamma}}\right)^{-1}$$

where, for conciseness, we let $\lambda_i \overset{\text{def}}{=} \lambda_i(\text{EGOP}(f))$ the ith eigenvalue of $\text{EGOP}(f)$, indexed in decreasing order. Expanding the above yields

$$\frac{\|\vec{\tilde{L}}\|_1}{\|\vec{L}\|_1} \leq d\left(\frac{\sqrt{\gamma}}{c} + HD_2\right) \cdot \frac{2c}{d}\frac{\sqrt{\lambda_1+\gamma}}{\beta_1\lambda_1-\gamma} + \frac{1}{c}\sum_{i=1}^{d}\sqrt{\lambda_i} \cdot \frac{2c}{d}\frac{\sqrt{\lambda_1+\gamma}}{\beta_1\lambda_1-\gamma} \tag{A.4}$$

$$= 2\left(\sqrt{\gamma} + HcD_2\right) \cdot \frac{\sqrt{\lambda_1+\gamma}}{\beta_1\lambda_1-\gamma} + 2\sum_{i=1}^{d}\sqrt{\lambda_i} \cdot \frac{1}{d}\frac{\sqrt{\lambda_1+\gamma}}{\beta_1\lambda_1-\gamma}. \tag{A.5}$$

Because $M_p \leq D_2^p$, by definition γ satisfies

$$\gamma \leq 2H\lambda_{\max}(\nabla^2 f(\theta^*))D_2^3 + H^2 D_2^4.$$

Thus

$$H \leq \frac{\sqrt{\delta\lambda_1(\text{EGOP}) + D_2^2\lambda_1^2(\nabla^2 f(\theta^*))} - D_2\lambda_1(\nabla^2 f(\theta^*))}{D_2^2},$$

implies $\gamma \leq \delta\lambda_1$ for $\lambda_1 \overset{\text{def}}{=} \lambda_1(\text{EGOP}(f))$, the largest EGOP eigenvalue. Additionally, H admits a simplified upper bound:

$$H \leq \frac{\sqrt{\delta\lambda_1(\text{EGOP}) + D_2^2\lambda_1^2(\nabla^2 f(\theta^*))} - D_2\lambda_1(\nabla^2 f(\theta^*))}{D_2^2} \leq \frac{\sqrt{\delta\lambda_1}}{D_2^2}.$$

Using these bounds on H and γ, the first term in Eq. A.5 can be bounded as

$$\begin{aligned} 2\left(\sqrt{\gamma} + HcD_2\right) \cdot \frac{\sqrt{\lambda_1+\gamma}}{\beta_1\lambda_1-\gamma} &\leq 2\left(\sqrt{\delta\lambda_1} + \frac{\sqrt{\delta\lambda_1}}{D_2^2} \cdot cD_2\right) \cdot \frac{\sqrt{1+\delta}}{\beta_1-\delta} \cdot \frac{1}{\sqrt{\lambda_1}} \\ &= 2\left(1 + \frac{cD_2}{D_2^2}\right)\frac{\sqrt{\delta(1+\delta)}}{\beta_1-\delta} \\ &\leq 6\frac{\sqrt{\delta(1+\delta)}}{\beta_1-\delta} \end{aligned}$$

where the last inequality follows from the fact that because $\text{supp}(\rho) \subseteq \Theta$, $c \leq D_2$.

The second term in Eq. A.5 can be bounded as

$$2\sum_{i=1}^{d}\sqrt{\lambda_i} \cdot \frac{1}{d}\frac{\sqrt{\lambda_1+\gamma}}{\beta_1\lambda_1-\gamma} \leq 2\sum_{i=1}^{d}\sqrt{\lambda_i} \cdot \frac{1}{d}\frac{\sqrt{1+\delta}}{(\beta_1-\delta)\sqrt{\lambda_1}} = \frac{2\,\text{sr}(f)}{d(\beta_1-\delta)}\sqrt{1+\delta}$$

where the equality follows from the definition of the stable rank. Combining these bounds yields the result. ∎

A.1.1. Convergence guarantees in the convex setting. Here we prove Theorem 3.1 by proving a more general statement in terms of any convex constraint set Θ.

Lemma A.8 (Convergence for convex objectives with noise-free gradients). *Consider $f(\cdot)$ a convex objective, constraint set Θ, and sampling distribution $\rho(\cdot)$ satisfying Assumptions*

1 and 2. Let $\tilde{f}(\cdot)$ and $\tilde{\Theta}$ denote the reparameterized objective and constraint set respectively corresponding to the eigenbasis of EGOP(f). *Then running Algorithm A.1 with constrained updates and with inputs $(\tilde{f}(\cdot), \nabla\tilde{f}(\cdot), \tilde{\theta}_0, T, \epsilon, \tilde{\Theta})$ for any initial condition $\tilde{\theta}_0 \in \tilde{\Theta}$ produces iterates $\{\tilde{\theta}_t\}_{t=1}^T$ satisfying*

(A.6)
$$\frac{1}{T}\sum_{t=0}^{T-1}(\tilde{f}(\tilde{\theta}_t) - \tilde{f}(\tilde{\theta}^*)) = O\left(\left(\eta + \frac{\tilde{D}_\infty^2}{\eta}\right)^2 \frac{\|\vec{L}\|_1}{T}\left(\frac{\mathrm{sr}(f)\sqrt{1+\delta}}{d(\beta_1-\delta)} + \frac{\sqrt{\delta(1+\delta)}}{\beta_1-\delta}\right) + \frac{\epsilon D_2^2}{\eta T}\right).$$

Setting $\eta = \tilde{D}_\infty$ to minimize this upper bound yields

$$\frac{1}{T}\sum_{t=0}^{T-1}(\tilde{f}(\tilde{\theta}_t) - \tilde{f}(\tilde{\theta}^*)) = O\left(\tilde{D}_\infty^2 \frac{\|\vec{L}\|_1}{T}\left(\frac{\mathrm{sr}(f)\sqrt{1+\delta}}{d(\beta_1-\delta)} + \frac{\sqrt{\delta(1+\delta)}}{\beta_1-\delta}\right) + \frac{\epsilon D_2^2}{\eta T}\right).$$

The proof follows by applying Thm. A.1 to an intermediate result in the proof of Thm. 4.1 in Liu et al. [25]. We defer a complete proof to Section E.

We note that the bound on reparameterized Adagrad is a function of $\tilde{D}_\infty$ while the bound on Adagrad in original coordinates is a function of D_∞. While $\tilde{D}_\infty^2$ can be smaller or larger than D_∞^2 by up to a factor of d, in many typical settings $\tilde{D}_\infty \leq D_\infty$: if the constraint set Θ is chosen to be a ball, $D_\infty = \tilde{D}_\infty$. Optimizing convex objectives with L2 regularization– sometimes referred to as with optimization with weight decay in machine learning literature– corresponds to optimizing with Θ some ball centered at the origin. In Section B.3, we present experiments using L2 regularization with adaptive algorithms, and confirm that the empirical benefit from reparameterization persists in this setting.

A.1.2. Convergence guarantees for non-convex objectives. In the non-convex setting we can establish related local improved convergence guarantee for unconstrained Adagrad. For a given initial point θ_0, let $\Delta_f(\theta_0) \stackrel{\text{def}}{=} f(\theta_0) - \min_\theta f(\theta)$. Observe that $\Delta_{\tilde{f}}(\tilde{\theta}_0) = \Delta_f(\theta_0)$ for $\tilde{\theta}_0 \stackrel{\text{def}}{=} V^T\theta_0$. For unconstrained optimization, we add the following additional assumption:

Assumption 3. *The objective $f(\cdot)$ is bounded below:* $\inf_\theta f(\theta) > -\infty$.

Theorem A.9 (Convergence for non-convex, unconstrained optimization). *Consider $f(\cdot)$, Θ, and $\rho(\cdot)$ satisfying Assumptions 1, 2, and 3. Let $\tilde{f}(\cdot)$ and $\tilde{\Theta}$ denote the reparameterization of $f(\cdot)$ and Θ by V the eigenbasis of* EGOP(f). *Consider any initialization $\theta_0 \in \Theta$, and let $\tilde{\theta}_0 \stackrel{\text{def}}{=} V^T\theta_0$. Assume $\exists\delta \in [0, \beta_1^2)$ such that $\nabla^2 f(\cdot)$ has Lipschitz constant H satisfying the bound in Eq. 3.2. Let $\{\tilde{\theta}\}_{t=1}^T$ denote the iterates produced by running Algorithm A.1 using unconstrained updates and with inputs $(\tilde{f}(\cdot), \nabla\tilde{f}(\cdot), \tilde{\theta}_0, T, \epsilon)$ for $\epsilon < 1/d$. Then for all T such that $\{\tilde{\theta}_t\}_{t=1}^T \subseteq \tilde{\Theta}$, we have*

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla f(\theta_t)\|_1 = O\left(\frac{\Delta_f(\theta_0)}{\eta\sqrt{T}} + \frac{\eta\|\vec{L}\|_1}{\sqrt{T}}\left(\frac{\mathrm{sr}(f)\sqrt{1+\delta}}{d(\beta_1-\delta)} + \frac{\sqrt{\delta(1+\delta)}}{\beta_1-\delta}\right)\log(p)\right)$$

where $p(T, \|\vec{\tilde{L}}\|_1, \nabla\tilde{f}(\theta_0)))$ is polynomial in T, $\|\vec{\tilde{L}}\|_1$, and $\|\nabla\tilde{f}(\tilde{\theta}_0)\|_\infty$.

We compare the result in Theorem A.9 to the convergence guarantee for Adagrad in original coordinates initialized at θ_0, as established in Jiang et al. [21]:

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla f(\theta_t)\|_1 = O\left(\frac{\Delta_f(\theta_0)}{\eta\sqrt{T}} + \frac{\eta\|\vec{L}\|_1}{\sqrt{T}}\log(p(T,\|\vec{L}\|_1,\nabla f(\theta_0)))\right).$$

As with Lemma A.8, this result implies that for $f(\cdot)$ with strong EGOP spectral decay and dense leading EGOP eigenvectors, the local convergence bounds for reparameterized can be smaller with by a factor of $1/d$.

We emphasize that the above is a *local* guarantee, and only holds for time horizons T such that $\{\tilde{\theta}_t\}_{t=1}^T \subseteq \tilde{\Theta}$. If iterates move arbitrarily far away from the region in which EGOP samples were concentrated, then without stronger assumptions about the landscape (i.e. vanishingly small H) one cannot hope that the EGOP reparameterization will be informative for new local geometry. For quadratic functions, $H = 0$ and thus the result holds for any T; we note that related works have restricted their analysis to quadratic functions for analyzing adaptive algorithms [42].

Proof of Thm. A.9. The result follows immediately from combining the convergence result in Thm. 3.1 of Jiang et al. [21] with the bound on $\|\vec{\tilde{L}}\|_1/\|\vec{L}\|_1$ from Thm. A.1. ∎

A.2. Proofs from Section 4. In Section 4, we noted that for objectives of the form $f(\theta) = h(A\theta)$, for some loss function $h(\cdot)$ and data matrix $A \in \mathbb{R}^{d\times n}$, the EGOP of $f(\cdot)$ is

$$\mathrm{EGOP}(f) = A^{\mathsf{T}}\mathbb{E}_{\theta\sim\rho}\left[\nabla_\theta h(A\theta)\nabla_\theta h(A\theta)^{\mathsf{T}}\right]A \tag{A.7}$$

For general loss functions $h(\cdot)$, one can establish the following upper bounds showing how the singular values of A control the EGOP eigepsectrum of $f(\cdot)$.

Lemma A.10. *Consider $f : \mathbb{R}^n \to \mathbb{R}$ satisfying $f(\theta) = h(A\theta)$ for some loss function $h : \mathbb{R}^n \to \mathbb{R}$ and nonsingular data matrix $A \in \mathbb{R}^{n\times n}$. Denote by $\sigma_i(\cdot)$ and $\lambda_i(\cdot)$ the ith singular value and eigenvalue of a matrix respectively, indexed by decreasing value. Then all nonzero eigenvalues of the EGOP of $f(\cdot)$ satisfy*

$$\frac{\lambda_k(\mathrm{EGOP}(f))}{\lambda_1(\mathrm{EGOP}(f))} \le \left(\frac{\sigma_k(A)}{\sigma_1(A)}\right)^2 \frac{\lambda_1(M)}{\lambda_n(M)}.$$

where

$$M \overset{\text{def}}{=} \mathbb{E}_{\theta\sim\rho}[\nabla_\theta h(A\theta)\nabla_\theta h(A\theta)^{\mathsf{T}}].$$

If M has some finite condition number that does not go to infinity as the spectral decay in A increases, then Lemma A.10 shows that increasing spectral decay in the data matrix A induces spectral decay in $\mathrm{EGOP}(f)$. For specific choices of $h(\cdot)$, one can more precisely characterize how spectral decay in the matrix A induces decay in the EGOP of $f \overset{\text{def}}{=} h \circ A$:

Lemma A.11. *Consider $A \in \mathbb{R}^{d\times n}$ and let $f(\theta) = \frac{1}{2}\|A\theta - y\|_2^2$ where $y = A\theta^* + \eta$, for η some mean-zero measurement noise. Assume sampling density ρ is a standard Gaussian distribution. Then the eigenvalues of* $\mathrm{EGOP}(f)$, $\{\lambda_k\}_{k=1}^d$ *indexed in decreasing order, satisfy*

$$\frac{\lambda_k}{\lambda_1} \le \left(\frac{\sigma_{k-1}(A)}{\sigma_1(A)}\right)^4 \quad \forall k \in [2,\dots,n] \tag{A.8}$$

683 *where $\sigma_i(A)$ denotes the i^{th} singular value of A, indexed in decreasing order.*

684 We defer the proof of Lemma A.11 to Section E.2.

685 ## REFERENCES

686 [1] E. Alpaydin and C. Kaynak, *Optical Recognition of Handwritten Digits*. UCI Machine Learning Repos-
687 itory, 1998. DOI: https://doi.org/10.24432/C50P49.
688 [2] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula,
689 A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, *JAX: composable transfor-*
690 *mations of Python+NumPy programs*, 2018, http://github.com/jax-ml/jax.
691 [3] C. Cartis, X. Liang, E. Massart, and A. Otemissov, *Learning the subspace of variation for global*
692 *optimization of functions with low effective dimension*, arXiv preprint arXiv:2401.17825, (2024).
693 [4] X. Chen and E. Hazan, *Open problem: Black-box reductions and adaptive gradient methods for noncon-*
694 *vex optimization*, in Proceedings of Thirty Seventh Conference on Learning Theory, S. Agrawal and
695 A. Roth, eds., vol. 247 of Proceedings of Machine Learning Research, PMLR, 30 Jun–03 Jul 2024,
696 pp. 5317–5324, https://proceedings.mlr.press/v247/chen24e.html.
697 [5] H.-H. Chou, C. Gieshoff, J. Maly, and H. Rauhut, *Gradient descent for deep matrix factorization:*
698 *Dynamics and implicit bias towards low rank*, Applied and Computational Harmonic Analysis, 68
699 (2024), p. 101595.
700 [6] P. G. Constantine, *Active subspaces: Emerging ideas for dimension reduction in parameter studies*,
701 SIAM, 2015.
702 [7] R. Cosson, A. Jadbabaie, A. Makur, A. Reisizadeh, and D. Shah, *Low-Rank Gradient Descent*,
703 IEEE Open Journal of Control Systems, (2023).
704 [8] B. Crew, *Google Scholar reveals its most influential papers for 2020*, Jul 2020.
705 [9] C. Cui, K. Zhang, T. Daulbaev, J. Gusak, I. Oseledets, and Z. Zhang, *Active subspace of neural*
706 *networks: Structural analysis and universal attacks*, SIAM Journal on Mathematics of Data Science,
707 2 (2020), pp. 1096–1122.
708 [10] DeepMind, I. Babuschkin, K. Baumli, A. Bell, S. Bhupatiraju, J. Bruce, P. Buchlovsky,
709 D. Budden, T. Cai, A. Clark, I. Danihelka, A. Dedieu, C. Fantacci, J. Godwin, C. Jones,
710 R. Hemsley, T. Hennigan, M. Hessel, S. Hou, S. Kapturowski, T. Keck, I. Kemaev,
711 M. King, M. Kunesch, L. Martens, H. Merzic, V. Mikulik, T. Norman, G. Papamakarios,
712 J. Quan, R. Ring, F. Ruiz, A. Sanchez, L. Sartran, R. Schneider, E. Sezener, S. Spencer,
713 S. Srinivasan, M. Stanojević, W. Stokowiec, L. Wang, G. Zhou, and F. Viola, *The DeepMind*
714 *JAX Ecosystem*, 2020, http://github.com/google-deepmind.
715 [11] A. Défossez, L. Bottou, F. Bach, and N. Usunier, *A Simple Convergence Proof of Adam and*
716 *Adagrad*, arXiv preprint arXiv:2003.02395, (2020).
717 [12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *Imagenet: A large-scale hierarchi-*
718 *cal image database*, in 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009,
719 pp. 248–255, https://doi.org/10.1109/CVPR.2009.5206848.
720 [13] Z. Dong, Y. Zhang, J. Yao, and R. Sun, *Towards quantifying the hessian structure of neural networks*,
721 arXiv preprint arXiv:2505.02809, (2025).
722 [14] J. Duchi, E. Hazan, and Y. Singer, *Adaptive subgradient methods for online learning and stochastic*
723 *optimization.*, Journal of Machine Learning Research, 12 (2011).
724 [15] X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, in
725 Proceedings of the thirteenth international conference on artificial intelligence and statistics, JMLR
726 Workshop and Conference Proceedings, 2010, pp. 249–256.
727 [16] G. H. Golub, *Some modified matrix eigenvalue problems*, SIAM Review, 15 (1973), pp. 318–334.
728 [17] V. Gupta, T. Koren, and Y. Singer, *Shampoo: Preconditioned stochastic tensor optimization*, in
729 International Conference on Machine Learning, PMLR, 2018, pp. 1842–1850.
730 [18] N. Halko, P.-G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic*
731 *algorithms for constructing approximate matrix decompositions*, SIAM Review, 53 (2011), pp. 217–
732 288.

[19] E. Hazan et al., *Introduction to online convex optimization*, Foundations and Trends® in Optimization, 2 (2016), pp. 157–325.

[20] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[21] R. Jiang, D. Maladkar, and A. Mokhtari, *Convergence analysis of adaptive gradient methods under refined smoothness and noise assumptions*, arXiv preprint arXiv:2406.04592, (2024).

[22] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017, https://arxiv.org/abs/1412.6980, https://arxiv.org/abs/1412.6980.

[23] F. Kunstner, R. Yadav, A. Milligan, M. Schmidt, and A. Bietti, *Heavy-tailed class imbalance and why Adam outperforms gradient descent on language models*, arXiv preprint arXiv:2402.19449, (2024).

[24] S. Z. Ling, N. Sharp, and A. Jacobson, *Vectoradam for rotation equivariant geometry optimization*, Advances in Neural Information Processing Systems, 35 (2022), pp. 4111–4122.

[25] Y. Liu, R. Pan, and T. Zhang, *AdaGrad under Anisotropic Smoothness*, arXiv preprint arXiv:2406.15244, (2024).

[26] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, 2019, https://arxiv.org/abs/1711.05101, https://arxiv.org/abs/1711.05101.

[27] L. Maes, T. H. Zhang, A. Jolicoeur-Martineau, I. Mitliagkas, D. Scieur, S. Lacoste-Julien, and C. Guille-Escuret, *Understanding Adam Requires Better Rotation Dependent Assumptions*, arXiv preprint arXiv:2410.19964, (2024).

[28] N. Mallinar, D. Beaglehole, L. Zhu, A. Radhakrishnan, P. Pandit, and M. Belkin, *Emergence in non-neural models: grokking modular arithmetic via average gradient outer product*, arXiv preprint arXiv:2407.20199, (2024).

[29] V. Papyan, *The full spectrum of deepnet hessians at scale: Dynamics with sgd training and sample size*, arXiv preprint arXiv:1811.07062, (2018).

[30] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, *Automatic differentiation in PyTorch*, in NIPS-W, 2017.

[31] A. Radhakrishnan, D. Beaglehole, P. Pandit, and M. Belkin, *Mechanism of feature learning in deep fully connected networks and kernel machines that recursively learn features*, arXiv preprint arXiv:2212.13881, (2022).

[32] S. J. Reddi, S. Kale, and S. Kumar, *On the Convergence of Adam and Beyond*, 2019, https://arxiv.org/abs/1904.09237, https://arxiv.org/abs/1904.09237.

[33] M. Rudelson and R. Vershynin, *Sampling from large matrices: An approach through geometric functional analysis*, Journal of the ACM (JACM), 54 (2007), pp. 21–es.

[34] L. Sagun, U. Evci, V. U. Guney, Y. Dauphin, and L. Bottou, *Empirical analysis of the hessian of over-parametrized neural networks*, arXiv preprint arXiv:1706.04454, (2017).

[35] M. Udell and A. Townsend, *Why are big data matrices approximately low rank?*, SIAM Journal on Mathematics of Data Science, 1 (2019), pp. 144–160.

[36] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*, Nature Methods, 17 (2020), pp. 261–272, https://doi.org/10.1038/s41592-019-0686-2.

[37] N. Vyas, D. Morwani, R. Zhao, I. Shapira, D. Brandfonbrener, L. Janson, and S. Kakade, *SOAP: Improving and stabilizing Shampoo using Adam*, arXiv preprint arXiv:2409.11321, (2024).

[38] R. Ward, X. Wu, and L. Bottou, *Adagrad stepsizes: Sharp convergence over nonconvex landscapes*, Journal of Machine Learning Research, 21 (2020), pp. 1–30.

[39] H. Xiao, K. Rasul, and R. Vollgraf, *Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms*, arXiv preprint arXiv:1708.07747, (2017).

[40] S. Xie, M. A. Mohamadi, and Z. Li, *Adam Exploits ℓ_∞-geometry of Loss Landscape via Coordinate-wise Adaptivity*, 2024, https://arxiv.org/abs/2410.08198, https://arxiv.org/abs/2410.08198.

[41] M. D. Zeiler, *Adadelta: An adaptive learning rate method*, 2012, https://arxiv.org/abs/1212.5701, https://arxiv.org/abs/1212.5701.
[42] Y. Zhang, C. Chen, T. Ding, Z. Li, R. Sun, and Z.-Q. Luo, *Why transformers need Adam: A Hessian perspective*, arXiv preprint arXiv:2402.16788, (2024).
[43] J. Zhao, Z. Zhang, B. Chen, Z. Wang, A. Anandkumar, and Y. Tian, *Galore: Memory-efficient llm training by gradient low-rank projection*, arXiv preprint arXiv:2403.03507, (2024).
[44] L. Zhu, C. Liu, A. Radhakrishnan, and M. Belkin, *Catapults in SGD: spikes in the training loss and their impact on generalization through feature learning*, arXiv preprint arXiv:2306.04815, (2023).

Supplementary material begins here.

Appendix B. Supplementary Figures.

B.1. Supplementary Figures for Section 4.

Spectral decay is robust to choice of sampling distribution. In Figure 10, we present evidence that spectral decay visualized in Figure 3 is robust to choice of sampling distribution. The EGOP whose eigenspectrum is displayed in Figure 3 was generated from gradient samples $\nabla f(\theta_i)$ for $\theta_i \sim \mathcal{N}(0, \mathbb{I})$, but as visualized in later figures (Figure 10), the level of spectral decay is comparable to that obtained with Gaussian distributions of differing scales. We also compare to the spectral decay exhibited by the EGOP matrix when ρ is taken to be an initialization distribution used in practice [15]. For details on this distribution, see Section C.6.

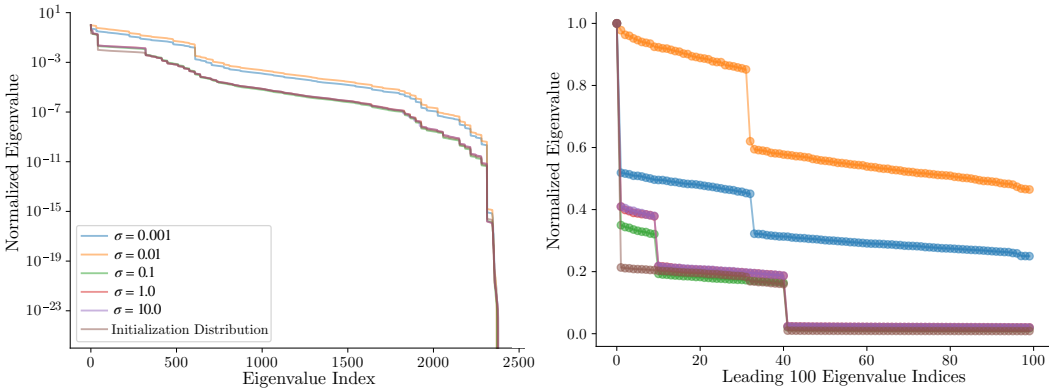


Figure 10: Comparing EGOP spectral decay of a 2-layer ReLU network on tinyMNIST dataset. Plot displays the ratio λ_k/λ_1 as a function of eigenvalue index k, for eigenvalues indexed in decreasing order. The blue, orange, green, red, and purple colored traces display the eigenspectrum of the EGOP with respect to a mean-zero Gaussian with covariance $\sigma^2\mathbb{I}$, for varying values of σ. The brown trace displays the eigenspectrum of the EGOP with respect to a realistic initialization distribution for this architecture: weights for each layer are drawn from a scaled Xavier normal distribution, and biases are initialized from a scaled uniform distribution (see Section C.6). We observe that under all sampling distributions, the eigenspectrum exhibits spectral decay, and that the realistic initialization distribution has spectral decay very comparable to that of the standard Gaussian, displayed in Figure 3 of the main body.

We also note that the interesting shelf structure of the leading eigenvalues is also robust to choice of sampling distribution, as illustrated in Figure 10. Full details for the objective and EGOP estimation procedure for this figure are detailed in Section C.5.

Spectral decay persists in block EGOP matrices. Figure 11 plots the normalized eigenspectra of the block matrices corresponding to the first and second layers of ReLU networks on the UCI digits dataset and fashionMNIST dataset respectively. For full details on these datasets and the architectures used, see Section C.5. Interestingly, both datasets exhibit shared characteristics: the normalized spectral decay in the first layer is strikingly similar, and in both networks the spectral decay in the first layer is more pronounced than in the second layer.

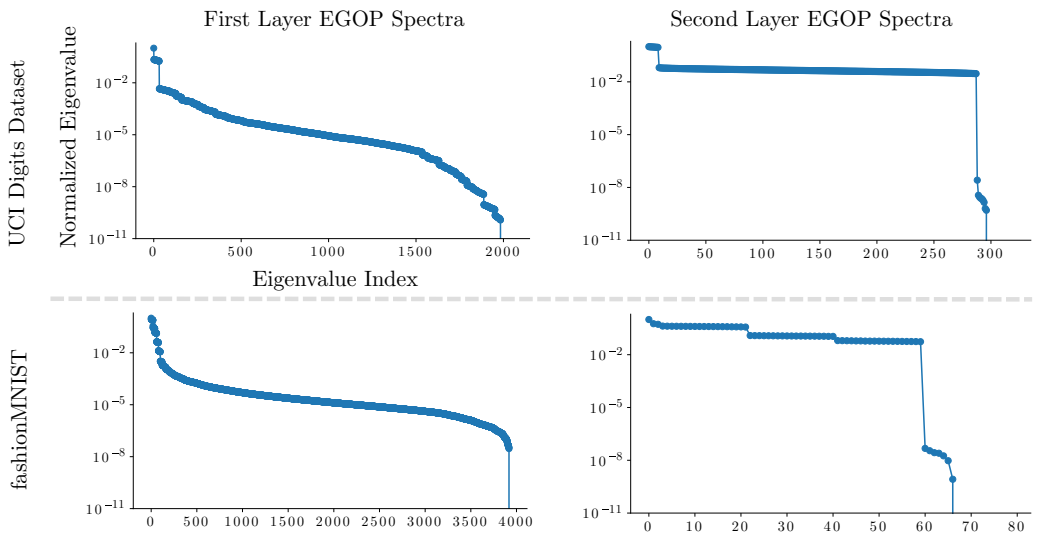


Figure 11: Eigenspectra of the layerwise EGOP matrices of neural networks on `tinyMNIST` and `fashionMNIST`. The spectral decay observed in Figure 3 persists for layer EGOP matrices, defined in Section 5, and across datasets. Y-axes for all figures display identical ranges.

Density of leading EGOP eigenvectors. Thms. A.8 and A.9 show that when the leading EGOP eigenvector is dense, reparameterized Adagrad enjoys much stronger convergence guarantees. Density of the k^{th} eigenvector is measured by $\beta_k \overset{\text{def}}{=} \|v_k\|_1^2$. Our results state a guarantee in terms of $\beta \overset{\text{def}}{=} \beta_1$, the density of the leading eigenvector.

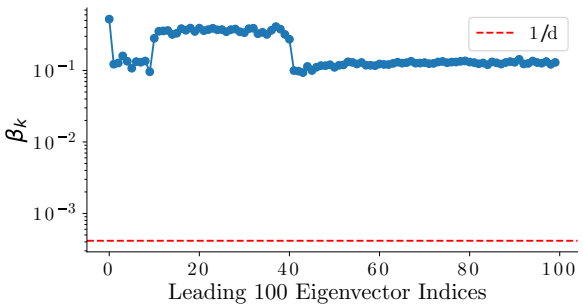


Figure 12: Plotting the density measure $\beta_k \overset{\text{def}}{=} \|v_k\|_1^2$ for the leading 100 eigenvectors of EGOP(f), where $f(\cdot)$ is the cross-entropy loss of a 2-layer ReLU neural network on the UCI digits training dataset. The leading eigenvector satisfies $\beta_1 > 0.5$ and several have density $\beta_k > 0.3$. We visualize the value $1/d$ in red (for this example, $d = 2,410$) to verify that for the leading eigenvectors, $\beta_k \gg 1/d$.

In order for the factors $\text{sr}(f)/d(\beta_1 - \delta)$ in the bound for reparameterized Adagrad from Thms. A.8 and A.9 to reflect an improvement, it is necessary that $\beta_1 \gg 1/d$. In Figure 12,

we show that for the UCI digits dataset, this condition is satisfied. Specifically, we show that $\beta_1 > 0.5$, while for this network $1/d < 5e-4$. Moreover, we show that not only does the leading eigenvector satisfy this density assumption, but several of the leading eigenvectors satisfy $\beta_k \gg 1/d$.

B.2. Supplementary Figures for Section 6.

Residual Networks. In Figure 13, we supplement the results shown in Figures 2 and 4. In Figure 13, we plot training and validation accuracy over epochs, confirming that the improved training loss convergence under reparameterization leads to improved classification accuracy.

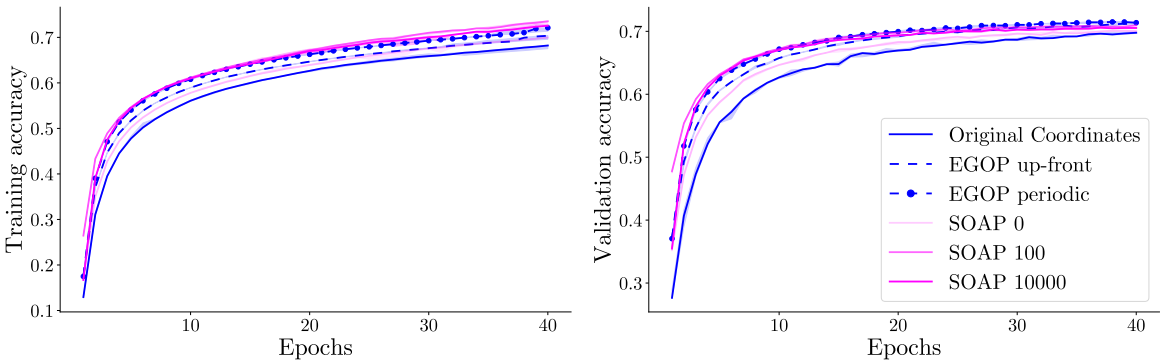
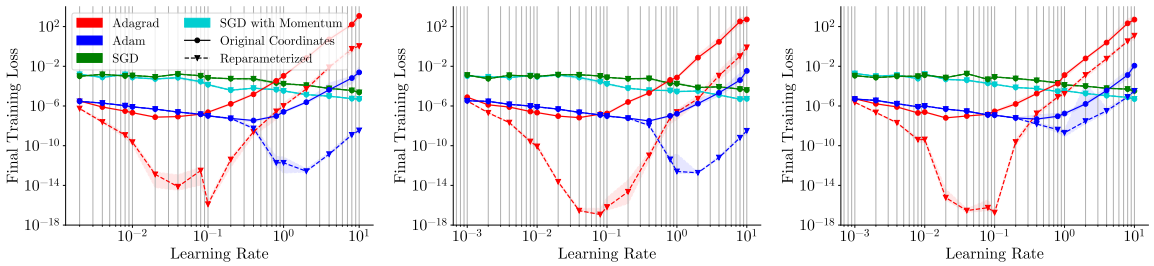


Figure 13: Training and validation accuracy for image classification with residual networks. The improved training and validation loss convergence displayed in Figures 2 and 4 leads to improved training and validation accuracy. EGOP reparameterization is competitive with SOAP in accuracy metrics as well as loss metrics.

Multilayer linear networks. We compare three methods for EGOP reparameterization in order to examine some heuristics proposed in Section 5. In Figure 14, we consider training a multilayer linear network (6.1) under global EGOP reparameterization, wherein all parameters are reparameterized simultaneously as in Algorithm 2.1 (Figure 14a); block reparameterization for all layers, following the procedure defined in Section 5 (Figure 14b); and block reparameterization of only the parameters in the first layer (Figure 14c). For all three methods, we estimate the EGOP using the same number of gradient samples: $M = 2d$, where d is the total number of network parameters.

Comparing Figure 14a with Figure 14b shows that for this problem, EGOP reparameterization offers comparable benefit when using block reparameterization as when using global reparameterization. This suggests that block EGOP reparameterization, which has a reduced computational cost compared to global EGOP reparameterization, may be an effective way to accelerate adaptive methods when problem instances are too large to permit global reparameterization. Reparameterizing only the first layer (Figure 14c) improves Adagrad's performance by a margin comparable to that of global and block reparameterization of all layers, but the benefit to Adam under reparameterization of only the first layer is much less pronounced.

Image Classification with ReLU Networks. Figure 15 expands on the results shown in Figure 1 (right) for 2-layer ReLU networks on the UCI digits dataset. Figure 15a plots final



(a) Global EGOP Reparameterization

(b) Block reparameterization, all layers

(c) First layer only

Figure 14: Comparing three EGOP reparameterization methods for training a multilayer linear network (6.1). Figure 14a is a reproduction of Figure 6b in the main body, and shows results for performing EGOP reparameterization of all parameters simultaneously. Figure 14b shows results when performing block EGOP reparameterization, where each network layer forms a block. Figure 14c shows results when block-reparameterizing only the parameters in the first layer.

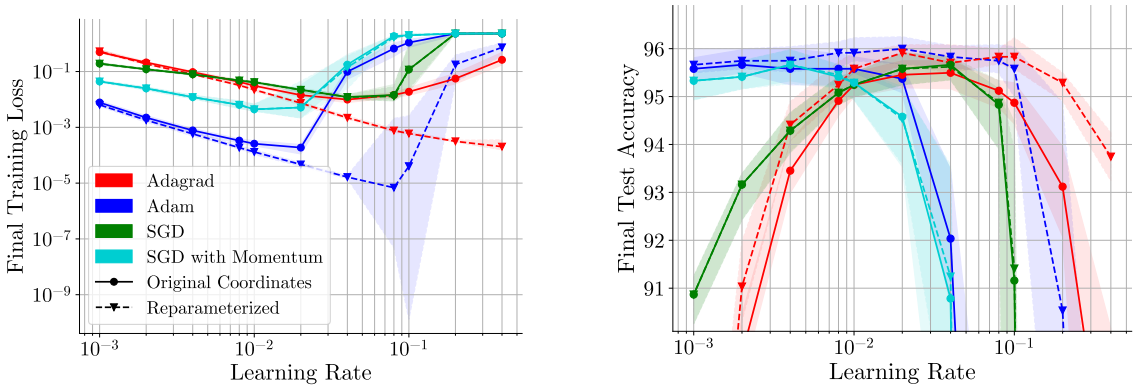
training loss versus learning rate, and shows that benefit of reparameterization shown in Figure 1 is robust to choice of learning rate. Figure 15b plots final test accuracy versus learning rate, and shows that the improved training offered by reparameterization does not lead to over-fitting. For full experimental details on the architecture, dataset, and loss function used for these experiments, see Section C.

Similarly, Figure 16 expands on the results presented in Figure 7. Figure 16 (left) shows that the improved training does not lead to over-fitting, but rather that reparameterization leads to improved accuracy on hold-out data. Figure 16 (right) reports wallclock time to convergence, demonstrating that the improved convergence in epochs shown in Figure 7 translates to faster convergence in wallclock time. For this experiment, we define convergence as the first epoch at which the model attains 99.5% of its maximum validation accuracy. In Figure 16 (left), we use square a circular markers to indicate the epochs at which the model in original coordinates and reparameterized models respectively achieve this threshold.

Convex Objectives. In addition to training neural networks, which is a primary application of adaptive optimization algorithms, we also study reparameterization for convex optimization. Figure 17a shows the result of minimizing

$$f(\theta) = \log\left(\sum_{i=1}^{n} \exp\left((\langle a_i, \theta\rangle - y_i)^2\right)\right), \tag{B.1}$$

where $a_i \in \mathbb{R}^d$ denote vectors of observations and $y_i \stackrel{\text{def}}{=} \langle a_i, \theta^*\rangle$ for ground truth θ^*. We also plot results from minimizing logistic regression objectives (Figure 17b) and linear least-squares objectives (Figure 17c) arising from problems with data matrices $A \in \mathbb{R}^{n\times d}$. For all convex objectives, we induce EGOP spectral decay by choosing matrices A with singular value decay. For these objectives, we estimate the EGOP and perform optimization using noiseless



(a) Median final training loss versus learning rate

(b) Median final test accuracy versus learning rate

Figure 15: Training loss and test accuracy results for the UCI digits dataset image classification task. Results are aggregated over independent trials corresponding to different random initializations. Medians are plotted as traces, and shaded regions indicate the 25^{th}-75^{th} percentiles.

(full-batch) gradients.

Figure 17 demonstrates that EGOP reparameterization can improve convergence of adaptive algorithms to global minima of convex objectives. EGOP-reparameterized Adagrad outperforms its counterpart in original coordinates for all three objectives. Notably, methods with momentum excel for this logistic regression, and reparameterization boosts the performance of Adagrad (which does not use momentum), making it competitive with the momentum-based optimizers. EGOP reparameterization improves Adam's convergence on the log-sum-exp objective and linear least-squares objectives (Figures 17a and 17c), but has no impact on Adam's performance for logistic regression (Figure 17b).

B.3. Optimizing with Weight Decay. In Section 3, we instantiated guarantees for a ball constraint set, and remarked that in practice adaptive optimization algorithms are often employed along with weight decay/L2 regularization, which implicitly constrains the algorithms to some ball in parameter space centered at the origin. We note that our experiments on residual networks and on image classification using fashionMNIST both employ AdamW, an adaptive algorithm with explicit weight decay, thus validating that EGOP reparameterization can also improve convergence when weights are implicitly constrained to some ball.

Appendix C. Experimental details.

Experiments were implemented in python, and both code and data will be made publicly available upon publication. Experiments with neural networks were implemented in Pytorch, and experiments with convex objectives were implemented using auto-differentiation in Jax and optimizers from Optax [30, 2, 10]. With the exception of experiments performed on residual networks, all experiments can be performed on CPU with a MacBook Pro with an

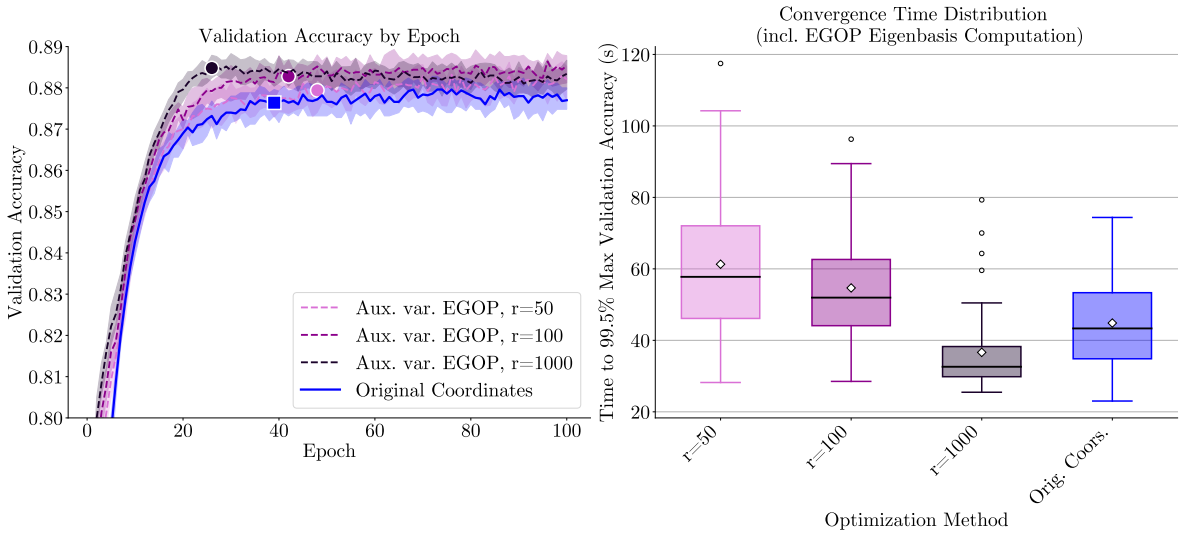


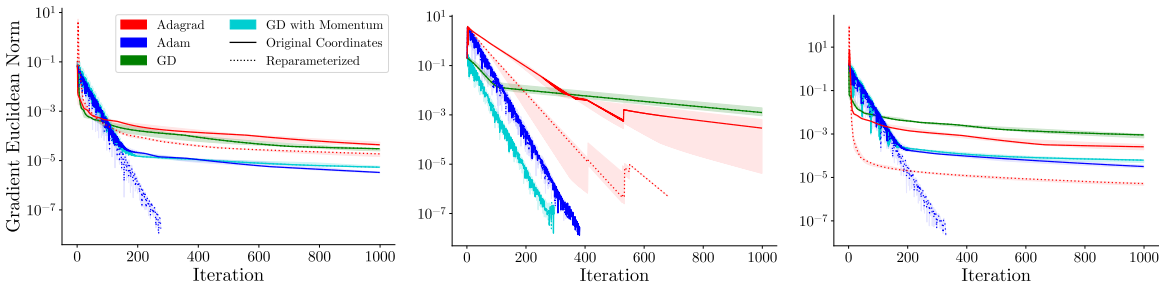
Figure 16: Validation accuracies and timing results for image classification on fashionMNIST. Counterpart to Figure 7. (Left) Plotting validation accuracy demonstrates that the improved minimization of training loss leads to improved generalization. (Right) Plotting wallclock time to convergence for the model in original coordinates, as well as under auxiliary variable EGOP reparameterization, as described in Section 5, for varying values of r. We find that the cost-benefit tradeoff favors intermediate values of r, e.g. $r = 1000$ in this figure. Recall that for this architecture, $r = 1000$ is much smaller than the number of optimization variables, $d \approx 78k$.

Intel Core i5 processor and 16GB memory. However, all experiments can be run more quickly on GPU, and GPU support is included in the codebase. Experiments with residual networks were performed on an internal cluster, using 1 RTX 6000 Ada GPU, and 8 CPU cores on an AMD EPYC 74F3 processor. The timing results with auxiliary variable reparameterization for ReLU networks in Figure 16 were performed on an internal cluster using 1 RTX A6000 GPU and 2 CPU cores on an Intel Xeon Silver 4114 processor.

C.1. Dataset Details.

UCI handwritten digits dataset. The UCI digits dataset [1] contains 5620 instances, each an 8×8 pixel greyscale image of a handwritten digit of values $0, \ldots, 9$. Each instance has a integer label, $0, \ldots, 9$. We split the dataset into a training dataset with 3823 instances, a validation dataset with 598 instances, and a test dataset with 1199 instances.

fashionMNIST dataset. The fashionMNIST dataset consists of 28×28 pixel grayscale images of clothing, each having a label from one of 10 classes. The full dataset contains 60k training samples and 10k test samples; for the results in Figure 7, we use the full training set and we subdivide the test set into a validation set of 2.5k samples and a test set of 7.5k samples [39]. For the experiments in Figure 11, we restrict to only instances corresponding to the first four classes (labeled "t-shirt," "trouser," "pullover," and "dress"). This yields



(a) Log-sum-exp with $\alpha = 2$ (b) Logistic regression with $\alpha = 3$ (c) Linear least-squares with $\alpha = 2$

Figure 17: Gradient Euclidean norm of solution at t^{th} iterate. Learning rates tuned separately for methods in original coordinates and under reparameterization. We induce EGOP spectral decay by choice of data matrix A with singular values $\sigma_k(A) = k^{-\alpha}$. As noted in the prose, in some plots the dotted traces coincide with the solid and are thus not visible (Adam in Figure 17b, and equivariant methods GD and GD with momentum).

a dataset of 28,000 instances, which we subdivide into a training set of 24,000 instances, a validation set of 1,000 instances, and a test set of 3,00 instances. We do this in order to reduce the problem size to a setting where we can visualize the full EGOP eigenspectrum obtained without using heuristics.

ImageNet dataset. For our large-scale experiments, we use the ImageNet classification dataset [12]. ImageNet consists of approximately 1.28 million training images and 50,000 validation images spanning 1,000 object categories. Following standard practice, we train on the full training set and report results on the validation set. During training, all models use the conventional preprocessing pipeline of random resized crops to 224×224 and horizontal flips. For evaluation, we resize each image so that its shorter side is 256 pixels and use a ten-crop evaluation protocol (four corners, center crop, and their horizontal flips). ImageNet serves as a challenging benchmark to evaluate the scalability and efficiency of our method in deep convolutional architectures.

C.2. Architecture Details.

Architecture for multilayer linear networks. To implement the objective

$$f(\theta) = \|W_3 W_2 W_1 A - Y\|_{\mathsf{F}}^2 / n_{\text{samples}} \tag{C.1}$$

we use a 3-layer fully connected network without any nonlinear activations. In accordance with the description in Section 6, the first layer has 10 input nodes and 30 output nodes, the second layer has 50 output nodes, and the last layer has 10 output nodes.

Architecture for UCI handwritten digits. For the UCI digits dataset we use a 2-layer ReLU network. Its first layer is a fully-connected linear layer with bias, with 64 input nodes and 32 output nodes, followed by a ReLU nonlinearity. Its second layer is a fully-connected linear layer with bias, with 32 input nodes and 10 output nodes, followed by a log-softmax. We form

the function $f(\cdot)$ by taking the negative log-likelihood loss on the training dataset.

Large architecture for fashionMNIST dataset. For the results in Figure 7, we use a 2-layer ReLU network. The first layer is a fully-connected linear layer with bias, with 784 input nodes and 100 output nodes, followed by a ReLU activation function. The second layer is a fully-connected linear layer with bias, with 100 input nodes and 10 output nodes. We form the function $f(\cdot)$ by taking the cross entropy loss on the training dataset. We note that this is equivalent to computing the log-softmax on the network output and then using negative log-likelihood loss, which is how we compute $f(\cdot)$ on tinyMNIST.

Small architecture for fashionMNIST dataset. Architecture details for auxilliary.

Architecture for ImageNet. For the ImageNet experiments, we use the 34 layer residual network (ResNet-34) architecture presented in the original paper [20]. The model begins with a convolutional layer consisting of a 7×7 convolution with 64 channels and stride 2, followed by batch normalization, a ReLU activation, and a 3×3 max pooling layer with stride 2. The network then contains four stages of residual blocks: the first stage has three residual blocks with 64 channels; the second stage has four residual blocks with 128 channels, with the first block using stride 2 for spatial downsampling; the third stage has six residual blocks with 256 channels, again with stride 2 in the first block; and the final stage has three residual blocks with 512 channels, with stride 2 in the first block. Each residual block consists of two 3×3 convolutional layers with batch normalization and ReLU activations, together with a skip connection. After the residual stages, we apply global average pooling and a fully connected layer mapping from 512 features to 1000 output classes. We define the function $f(\cdot)$ as the cross entropy loss computed on the ImageNet training dataset.

C.3. Algorithm Details. We consider four optimization algorithms: Adagrad, Adam, SGD, and SGD with momentum. We use standard settings for all hyperparameters except learning rate: for Adam, we use $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and for SGD with momentum we use momentum parameter 0.9, matching that of Adam. We use zero weight decay for all algorithms, and for all objectives except the linear feed-forward networks, we use a constant learning rate schedule.

We tune learning rates for each algorithm. Unless otherwise noted, when choosing the range of learning rates to sweep for tuning, we the following two-step procedure. First we sweep a coarse sweep using powers of 10 (e.g. 1e-3, 1e-2, ... , 100) to find an upper and lower bound on the learning rates that produce best performance. The metric by which we quantify the "best performance" for each different experiment is discussed below in Section C.6. We then perform a refined sweep, where we discretize each interval between subsequent power of 10 using doubling. For example, if the coarse sweep identified lower bound 0.01 and upper bound 1.0, we would perform the refined sweep over values [0.01, 0.02, 0.04, 0.08, 0.1, 0.2, 0.4, 0.8, 1.0].

Auxiliary Variables. Explain that we expect best results when we use weight decay because we're adding overparameterization i.e. auxiliary variables. Explain what values we used in our weight decay sweep.

ResNet for ImageNet. Following the standard ResNet training protocol, we use hyperparameters comparable to those reported in the original work. Although the original ResNet work [20] trains using SGD with momentum 0.9, our experiments use AdamW. To preserve

comparable smoothing of gradient information, we set $\beta_1 = 0.9$, which plays a role analogous to momentum in SGD, together with the standard choice $\beta_2 = 0.999$. We use the same weight decay parameter as the original ResNet work, namely 10^{-4}. For learning rate selection, we perform a logarithmic sweep over the interval $[10^{-5}, 1]$, training each candidate learning rate for 10 epochs and selecting the value that achieves the highest top-1 validation accuracy under ten-crop evaluation. After selecting the learning rate, we train the final model for 40 epochs using a batch size of 128 and the standard ImageNet data augmentations described in Section C.1. We chose 40 epochs because, under a fixed learning rate without decay, the validation losses consistently plateau by this point, indicating that the model has reached its first stage of convergence.

To compute the EGOP matrix for convolutional layers, we repeatedly reinitialize the network weights and record the gradients obtained from a single forward–backward pass at initialization. Because a single pass produces one gradient sample for each kernel in a convolutional layer, a layer with c output channels yields c gradient samples per pass. We therefore determine the required number of passes by ensuring that every convolutional layer accumulates at least 1000 gradient samples in total. Let $c_{\min}$ denote the smallest number of output channels across all convolutional layers. Since this layer contributes only $c_{\min}$ samples per pass, we perform $\lceil 1000/c_{\min} \rceil$ forward–backward passes. In each pass, we compute the gradient of the cross-entropy loss with respect to all convolutional and linear weights.

C.4. Additional Details for Figures in Section 1. For Figure 1 (left), we generate the loss landscape pictured by taking $d = 2$ dimensions and $n = 100$ samples. We generate a matrix A with singular values $\sigma_k = k^{-\alpha}$ with $\alpha = 1.5$ and random right- and left-singular vectors. We sampled $\theta^* \sim \mathcal{N}(0, \mathbb{I})$ and used A, θ^* to induce $f(\cdot)$ a log-sum-exp objective, as defined in Section 6. We generated problem instances at random, and chose a random seed that produced a problem where the primary directions of variation for $f(\cdot)$ were clearly un-aligned with the coordinate axes. We allow Adagrad to take 1000 iterations.

We selected an initial point θ_0 that would produce a clear visual distinction between the two coordinate systems in early iterates. To select the learning rate, we first swept over the values $1, 10, 20, \ldots, 100$, and examined the suboptimality of the solution produced after 1000 iterates. For each learning rate we conducted 5 random trials on log-sum-exp objectives generated with identical values of d, n, α, and with $\theta^* \sim \mathcal{N}(0, \mathbb{I})$; these trials were initialized at points drawn from $\mathbf{N}(0, \|\theta_0\|_2^2 \mathbb{I})$, where θ_0 is the initial point used in Figure 1 (left). We found that for learning rates 30 through 90, the suboptimality of the solution returned by the algorithm in original coordinates was very comparable across learning rates and very close to zero ($< 1e-10$). Thus we selected a learning rate with these properties that was on the lower end (hence 30) because at larger learning rates, the oscillations around the global minimum made it more difficult to visually assess the difference between trajectories before and after reparameterization.

C.5. Additional Details for Figures in Section 4. Figure 3 plots the eigenspectrum of an empirically estimated EGOP matrix. We use the UCI digits dataset and consider the training subset described in Section C.1. We use the 2-layer ReLU network detailed in Section C.2.

For Figure 3 we use full-batch gradients to estimate the EGOP matrix. We take ρ to be a standard normal distribution: for each gradient sample, we form θ by sampling the

entries of all weights and biases i.i.d. from a standard Gaussian. We estimate EGOP(f) using $M = 10d$, where d is the total number of parameters (sum of all weight and bias entries) for the network. This is a larger number of EGOP samples than we use in later experiments; this is done intentionally with the goal of clearly resolving the spectral decay, rather than having decay appear as an artifact of numerical estimation with few samples.

For Figure 10, we use the same dataset, architecture, objective $f(\cdot)$, and procedure for EGOP estimation, but we use different sampling distributions ρ. The "realistic initialization" distribution is described in full detail below in Section C.6.

For Figure 11, we use the same architecture and objective for the UCI digits dataset. For fashionMNIST, we use the train-validation-test split, as described in Section C.1, and use the architecture described in Section C.2.

For Figure 11, we use minibatches of size 300 to estimate gradients. For each architecture, we use $M = 5d$ gradient samples to estimate each block EGOP matrix, where d is the number of parameters in the network. This is a larger number of EGOP samples than we use in later experiments; this is done intentionally with the goal of clearly resolving the spectral decay, rather than having decay appear as an artifact of numerical estimation with few samples. We perform block EGOP reparameterization for both networks, where the blocks are defined by the weight matrices of each network. For more details on block reparameterization for neural network weights, see the example in Section D.

When drawing gradients to estimate the block EGOP matrices, the distribution over parameters ρ from which we draw is the same as the distribution we later use to initialize the networks during training. For a full description of this initialization distribution, see Section C.6.

C.6. Additional Details for Figures in Section 6.

Linear Feedforward Networks. As described in Section 6, for Figure 6 we consider parameters $\theta = [\text{vec}(W_1), \text{vec}(W_2), \text{vec}(W_3)]$ where $W_1 \in \mathbb{R}^{50\times 10}$, $W_2 \in \mathbb{R}^{30\times 50}$, $W_3 \in \mathbb{R}^{10\times 30}$. We seek to minimize loss

$$f(\theta) = \|W_3W_2W_1A - Y\|_{\mathsf{F}}^2/n_{\text{samples}}$$

where $A \in \mathbb{R}^{10\times n_{\text{samples}}}$, and $Y = M^*A$ for $M^* \in \mathbb{R}^{10\times 10}$ drawn from a standard Gaussian distribution. We induce spectral decay in EGOP(f) by generating A with singular values $\sigma_k(A) = k^{-2}$ and random right- and left-singular vectors. For each trial, we generate $20,000$ data samples, which we split into a test set of $10,000$ training data instances, $4,000$ validation instances, and $6,000$ test instances. We define the training loss to be $f(\cdot)$ restricted to the training instances and their labels, and use this function when estimating the EGOP matrix. We use stochastic mini-batches of size 500 when estimating the EGOP and when performing optimization.

For each trial, we generate A, M^*, and Y as described above. We then estimate EGOP(f) using $M = 2d$ samples, where d is the total number of parameters in the network ($d = 2,300$). When estimating the EGOP, we let ρ be the same distribution used when initializing networks for training. Specifically, we initialize each weight matrix from a mean-zero Xavier normal distribution, also called Glorot initialization, which is widespread in practice [15]. The Xavier normal distribution is a Gaussian with standard deviation $\sqrt{2/\text{fan-in-fan-out}}$, where for a matrix $W \in \mathbb{R}^{n\times m}$, fan-in-fan-out $= n + m$. We compute the full eigenvalue decomposition to

find the change-of-basis matrix V.

We use 1000 epochs during training. See Section C.3 for choice of hyperparameters and choice of the range of learning rates to sweep for tuning. We measure performance using the median minimum validation loss achieved during training. We perform 10 independent trials at each learning rate. For each algorithm, we choose the learning rate at which the algorithm in original achieved lowest median minimum validation loss. Results of this learning rate sweep are in Figure 6c.

Because Adam in original coordinates exhibited numerical instability due to the near-zero gradient values encountered, we used cosine annealing to decay the learning rate over the coarse of training. We use Pytorch's default implementation of cosine annealing. For ease of comparison, we apply the same learning rate decay schedule to all algorithms (Adagrad, Adam, SGD, and SGD with momentum) in both original and reparameterized coordinates. We use this learning rate decay schedule throughout learning rate tuning, and in all results displayed in Figure 6.

ReLU Networks for Image Classification. We use the UCI digits dataset and fashionMNIST, with the train-validation-split described in Section C.1 and architectures described in Section C.2. We use the same (random) partition of datapoints into train, validation, and test datasets for all trials.

For both networks and objectives, we use minibatches of size 300 to estimate gradients during both EGOP estimation and optimization. We perform block EGOP reparameterization for both networks, where the blocks are defined by the weight matrices of each network. For more details on block reparameterization for neural network weights, see the example in Section D. For the UCI digits dataset, we use $M = d$ minibatch gradient samples to estimate the EGOP, where d is the number of parameters in the network, and compute the orthonormal reparameterization matrices for each layer by taking the (full, deterministic) SVD of the matrix of gradient samples for that layer, as this is equivalent to taking the eigenvalue decomposition of the empirical block EGOP matrix.

For the fashionMNIST experiments reported in Figure 7, we use pytorch's `svd_lowrank` function to implement randomized SVD, which in turn is based off Algorithms 4.1 and 5.1 in Halko et al. [18]. This yields $V_r \in \mathbb{R}^{d_1 \cdot d_2 \times r}$ a matrix with orthonormal columns. We use this matrix to perform auxiliary variable reparameterization, as introduced in Section 5. See Section D for full details. The following are OLD experimental details. Update to describe number of EGOP samples for FashionMNIST. Keep use of randomized SVD discussion. For fashionMNIST, we collect $M = 0.1d$ minibatch gradient samples, where d is the number of parameters in the model. For each layer, of dimension $d_1 \times d_2$, we approximate the leading $0.1d_1 \cdot d_2$ left-singular vectors of the matrix of gradient samples for that layer.

For each objective and network, we use the same distribution over parameters for drawing gradient samples to estimate the EGOP and for initializing the network at training time. For the experiments in Figure 1 (right), we draw weight entries i.i.d. from a mean-zero Xavier distribution, and bias entries i.i.d. from a uniform distribution with range $[-(\text{fan-in-fan-out})^{-1/2}, (\text{fan-in-fan-out})^{-1/2}]$.

Check that these details about initialization and sampling distributions are accurate for new auxilliary method. If not, update. For the experiments in Figure 7, that distribution is Pytorch's default method for re-setting parameters in a linear layer, namely drawing weight

entries i.i.d. from a uniform distribution with range

$$[-(\text{in-features})^{-1/2}, (\text{in-features})^{-1/2}]$$

and drawing bias entries i.i.d. from a uniform distribution with range

$$[-(\text{fan-in-fan-out})^{-1/2}, (\text{fan-in-fan-out})^{-1/2}],$$

where fan-in-fan-out for a layer is defined above in Section C.5. All of these initialization distributions stem from heuristics that are widespread in practice, and the distinction between the weight distributions used with the digits dataset versus the fashionMNIST dataset was the result of using legacy code; we did not tune initializations to different applications.

We selected hyperparameters and tuned learning rates following the procedures discussed in Section C.3. We used a constant learning rate for both datasets, as all methods appeared stable. We measure performance using the median maximum validation classification accuracy achieved during training. For each algorithm, we choose the learning rate at which the algorithm in original achieved highest median maximum validation accuracy. For the digits dataset, we performed 50 independent trials at each learning rate. Check and update parameter tuning description for fashionMNIST if needed. For fashionMNIST, we performed a single trial at each learning rate, and only trained for 20 epochs during tuning. Once learning rates were selected, we set the number of epochs over which to train by examining loss and accuracy curves by eye, and choosing a value large enough that all algorithms had converged. For the digits dataset, we trained for 200 epochs, and for fashionMNIST we trained for 80 epochs.

Regression with Errors-in-Variables. As described in Section 6, we optimize

$$f(\theta) = \frac{1}{2}\|A(M \odot W) - Y\|_F^2.$$

For each trial, the data are generated as follows. We set the problem dimension to $n = d = 10$ and sample the design matrix A so that its singular values obey $\sigma_k = k^{-\alpha}$ with $\alpha = 2.0$. For $\nu \geq 0$, each entry M_{ij} is drawn independently from $\mathcal{N}(1, \nu^2)$. The ground truth parameter W^* is sampled entrywise from a standard normal distribution, and the target Y is set as $Y = A(M \odot W^*)$.

For optimization, we compare the performance of SOAP, Shampoo, base Adam and Adagrad, and EGOP-reparameterized Adam and Adagrad. All parameters are initialized at zero, and each optimization run proceeds for 1000 steps using a cosine annealing learning rate schedule. We employ this learning rate schedule in order to improve SOAP's convergence. On independent trials, a new random instance of A, M, W^*, and Y is generated, on which each optimizer is evaluated.

We perform a logarithmic learning rate sweep for each optimizer and each ν. In every trial, the training loss trajectory is recorded. We measure performance for each learning rate using the median final training loss across the five trials. The best learning rate for each setting is taken to be the one with the lowest median final loss.

For EGOP reparameterization, we sample 1000 gradients from parameters W, where the entries of W are sampled from a standard normal distribution.

Low-rank Matrix Factorization. As stated in Section 6, we consider minimizing

$$f(\theta) = \frac{1}{2}\|A(LR^T) - Y\|_F^2.$$

where $L \in \mathbb{R}^{n\times r}$ and $R \in \mathbb{R}^{m\times r}$, and $A \in \mathbb{R}^{n\times n}$. On each trial, the synthetic ground truth X^* is generated to have rank r and a prescribed condition number κ. Specifically, we factor X^* as $X^* = U\Sigma V^\top$ with random orthonormal matrices $U \in \mathbb{R}^{n\times r}$, $V \in \mathbb{R}^{m\times r}$, and singular values $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$, for values forming a linear grid from 1 to $1/\kappa$. The measurement operator A is also constructed as a random matrix with controlled singular value decay: its singular values follow $\sigma_k = k^{-\alpha}$. Both X^* and A are independently regenerated for each trial. Each observation matrix Y was generated using $Y = AX^* + \mathcal{N}(0, \sigma_{\text{obs}}^2)$. All experiments are performed with $n = m = 20$, $r = 5$, $\kappa = 2$, and $\alpha = 1/2$, and the standard deviation of the observation noise was set to $\sigma_{\text{obs}} = 0.01$.

We use spectral initialization: to form L_0, R_0, we solve for $A^\dagger Y$ where $A^\dagger$ indicates the Moore-Penrose pseudoinverse, perform an SVD, and truncate to rank r to obtain U_r, Σ_r, V_r, and form left- and right- factors $L_0 = U_r\Sigma_r{-}1/2$, $R_0 = V_r\Sigma_r^{-1/2}$. SOAP, Shampoo, and base Adam and Adagrad are initialized at (L_0, R_0), and EGOP-reparameterized methods are initialized at the appropriate equivalent points in the coordinates of the EGOP-eigenbasis.

For each optimizer, we perform logarithmic learning rate sweeps, performing five independent trials on each. For each optimizer, we selected the learning rate achieving the lowest median final loss. All parameters are optimized for 1000 steps starting from this initialization. Figure 9 reports results aggregated over 5 independent trials, corresponding to problem instances *independent* from any problems generated during the learning rate sweep.

Convex objectives. We study three objectives. For each, we generate a matrix $A \in \mathbb{R}^{n_{\text{samples}}\times d}$ with singular value decay $\sigma_k = k^{-\alpha}$ and random orthonormal right- and left-singular vectors. The values of α are specified in the caption of Figure 17. The log-sum-exp objective is defined in Eq. B.1. The logistic regression objective is defined as

$$f(\theta) = \sum_{i=1}^{n_{\text{samples}}} -y_i \log\left(\frac{1}{1+e^{-\langle a_i,\theta\rangle}}\right) - (1-y_i)\log\left(\frac{e^{-\langle a_i,\theta\rangle}}{1+e^{-\langle a_i,\theta\rangle}}\right)$$

where $\{a_i \in \mathbb{R}^d\}_{i=1}^{n_{\text{samples}}}$ are the columns of A, and labels $y_i \sim \text{Bernoulli}(\pi(\theta^*)_i)$ where $\theta^* \in \mathbb{R}^d$ is drawn from $\mathcal{N}(0, \mathbb{I})$ and

$$\pi(\theta^*)_i \stackrel{\text{def}}{=} \frac{e^{-\langle a_i,\theta^*\rangle}}{1+e^{-\langle a_i,\theta^*\rangle}}.$$

The logistic regression objective is

$$f(\theta) = \frac{1}{2}\|A\theta - y\|_2^2$$

where $y = A\theta^*$ and $\theta^* \in \mathbb{R}^d$ is drawn from $\mathcal{N}(0, \mathbb{I})$.

For log-sum-exp and linear least squares, $n_{\text{samples}} = d = 100$. For logistic regression, $n_{\text{samples}} = 100$ and $d = 3$. For each objective, on each trial we generate A, θ^*, and thus $f(\cdot)$ randomly and independently following the above procedure.

For all convex objectives, use deterministic (full-batch) gradients to estimate the EGOP and to optimize. For all objectives, we use $M = 5d$ gradient samples to estimate the EGOP matrix. We take ρ to be a standard Gaussian distribution, and use the same distribution to initialize when optimizing. See Section C.3 for choice of hyperparameters and choice of the range of learning rates to sweep for tuning. We used a constant learning rate for all convex objectives. We measure performance using the median final training loss achieved. For each algorithm, we choose the learning rate at which the algorithm in original achieved lowest median training loss. We perform 5 independent trials at each learning rate.

For each objective, we optimize for 1000 iterations, or until the gradient Euclidean norm drops below $1e-8$. The latter termination condition is comparable to default termination conditions employed by `scipy.optimize` [36].

Appendix D. Expanded Discussion of Heuristics for Scalability. In this section, we expand on the heuristics proposed in Section 5.

Block Reparameterization. Here we instantiate block reparameterization for multilayer neural networks. Given an L-layer neural network whose ith layer is parameterized by weight matrix $W_i \in \mathbb{R}^{n_{\text{in}}^{(i)} \times n_{\text{out}}^{(i)}}$ and bias vector $b_i \in \mathbb{R}^{n_{\text{out}}^{(i)}}$, we consider optimizing $f(\theta) = \text{loss}(\theta; X_{\text{train}}, y_{\text{train}})$ with respect to parameters $\theta = [(W_1, b_1), \ldots, (W_L, b_L)]$.

Rather than forming the full EGOP as described in Algorithm 2.1, we consider estimating the *layer* EGOP matrices

$$\text{EGOP}^{(i)} \overset{\text{def}}{=} \frac{1}{M} \sum_{k=1}^{M} \nabla_{W_i} f(\theta_k) \nabla_{W_i} f(\theta_k)^{\mathsf{T}}$$

where $\nabla_{W_i} f(\theta_k) \in \mathbb{R}^{n_{\text{in}}^{(i)} n_{\text{out}}^{(i)}}$ is the vector of partial derivatives of f w.r.t. the entries of W_i evaluated at θ_k, and the points $\{\theta_k\}_{k=1}^{M} \sim \rho$ are drawn i.i.d..

Given these L empirical layer EGOP matrices, one can obtain L change-of-basis matrices $V^{(i)} \in \mathbb{R}^{n_{\text{in}}^{(i)} n_{\text{out}}^{(i)} \times n_{\text{in}}^{(i)} n_{\text{out}}^{(i)}}$, and form the objective

$$\tilde{f}(\theta) = f([(V^{(1)} W_1, b_1), \ldots, (V^{(L)} W_L, b_L)]).$$

Block EGOP reparameterization requires storing and applying L orthonormal matrices, each of size $n_{\text{in}}^{(i)} n_{\text{out}}^{(i)} \times n_{\text{in}}^{(i)} n_{\text{out}}^{(i)}$. For deep neural networks, this can be considerably less expensive than storing and applying the global change-of-basis matrix, which would be of size $\sum_{i=1}^{L} n_{\text{in}}^{(i)} n_{\text{out}}^{(i)} \times \sum_{i=1}^{L} n_{\text{in}}^{(i)} n_{\text{out}}^{(i)}$. This may also reduce the sampling cost; each layer EGOP is of smaller dimension than the global EGOP and thus has a smaller number of eigenvectors to estimate. Thus the layer EGOP eigenbases may be estimated with fewer total gradient samples.

Another benefit to block EGOP reparameterization is that it offers an easy way to reduce cost by choosing to only reparameterize a subset of layers. Thus for example if a network has a subset of layers which are too wide to efficiently reparameterize, one can choose to only reparameterize the narrower layers.

For networks where the first layer involves a linear transformation of the input data, reparameterizing only the first layer corresponds to an orthogonal transformation of the data.

Thus pre-computing the EGOP eigenbasis and applying this matrix to the input data up-front would allow one to reparameterize the first layer, without requiring one to store and apply the change-of-basis matrix during training.

D.1. Efficient Heuristics for Large-Scale Neural Networks. One major application area for adaptive gradient methods is training of massive multi-layer neural networks. In this section, we present computational heuristics that enable EGOP-reparameterization to scale to massive architectures.

[AD : Rick and Jose: you can both access commands that allow you to leave notes like this. You can change your color by going to TMLR/TMLR_custom_commands.sty and editing your command.] **Jose**: This is an example of Jose's note command **Rick**: This is an example of Rick's note command

D.1.1. Reparameterizing Massive Fully-Connected Layers. For fully-connected layers with a large number of parameters, storing and applying the full reparameterization matrix $V \in \mathbb{R}^{d\times d}$ can be prohibitively expensive. We describe two memory-efficient variants that compute and retain only the top-r eigenvectors of the EGOP matrix, forming a reduced basis $V_r \in \mathbb{R}^{d\times r}$ where $r \ll d$. The first approach constrains parameters to the subspace spanned by V_r, while the second uses auxiliary variables to maintain full expressivity while still storing only V_r.

Reduced-dimension EGOP. The primary motivation for reduced-dimension reparameterization is to reduce the memory cost of storing the reparameterization matrix V. Storing the full orthonormal basis $V \in \mathbb{R}^{d\times d}$ requires $O(d^2)$ memory, while storing only the top-r eigenvectors $V_r \in \mathbb{R}^{d\times r}$ requires $O(dr)$ memory. Since the memory savings from reducing V scale as $d^2 - dr = d(d-r)$ while the savings from reducing the parameter count scale as $d-r$, shrinking V to V_r provides a factor of d larger benefit than shrinking θ to θ_r. Additionally, computing only the top-r eigenvectors via randomized SVD is significantly faster than computing the full eigendecomposition.

Rather than the full reparameterization $f(\theta) = f(V\theta)$ with $\theta \in \mathbb{R}^d$ and $V \in \mathbb{R}^{d\times d}$, reduced-dimension EGOP uses

$$f(\theta) = f(V_r\theta_r)$$

where $\theta_r \in \mathbb{R}^r$ and $V_r \in \mathbb{R}^{d\times r}$ contains the top-r eigenvectors of the empirical EGOP matrix $\widehat{P}$. This constrains θ to lie in the r-dimensional subspace $\text{span}(V_r) \subset \mathbb{R}^d$.

Applicability and limitations. In practice, we compute V_r by first sampling M gradients $\{g_i\}_{i=1}^M$ at random initializations $\{\theta_i\}_{i=1}^M \sim \rho$, forming the gradient matrix $G \in \mathbb{R}^{d\times M}$ with columns g_i. We then apply randomized SVD to G to obtain the top-r left singular vectors, which form V_r. This approach avoids forming the full $\widehat{P} = \frac{1}{M}GG^\mathsf{T}$ matrix and is computationally cheaper than full eigendecomposition.

Computing the reduced basis. If the EGOP matrix V has exact rank r, then V_r spans the full column space of V and we lose no expressivity by switching to V_r. However, in practice, the EGOP matrix is typically full-rank with rapidly decaying eigenvalues rather than having exact rank r. Thus reduced-dimension reparameterization necessarily introduces a projection error of $\|\theta - V_rV_r^\mathsf{T}\theta\| > 0$ for general $\theta \in \mathbb{R}^d$, limiting the expressivity of the reparameterized model to the r-dimensional subspace $\text{span}(V_r)$. We introduce auxiliary variables EGOP below

to overcome this limitation.

Auxiliary variables EGOP. Rather than completing the basis by appending a random orthonormal basis for $\text{span}(V_r)^\perp$, we introduce auxiliary parameters to represent the orthogonal complement. The key idea is to decompose parameters into two orthogonal components via the reparameterized objective:

$$\tilde{f}(\theta_r, \theta_d) = f(V_r\theta_r + (I - V_rV_r^T)\theta_d)$$

where $\theta_r \in \mathbb{R}^r$ represents coordinates in the dominant r-dimensional EGOP subspace, and $\theta_d \in \mathbb{R}^d$ are auxiliary variables whose projection $(I - V_rV_r^\mathsf{T})\theta_d$ represents the residual component orthogonal to this subspace. The full parameter vector in $\mathbb{R}^d$ is thus reconstructed as:

$$\theta = V_r\theta_r + (I - V_rV_r^T)\theta_d$$

where $V_rV_r^T$ is the orthogonal projection onto $\text{span}(V_r)$. The explicit projection $(I - V_rV_r^T)$ applied to θ_d ensures the residual component remains in $\text{span}(V_r)^\perp$ throughout optimization. Although the parameter count increases from r to $r + d$ relative to the reduced-dimension approach, memory cost remains dominated by V_r storage (which scales as $O(dr)$) rather than parameter storage (which scales as $O(d)$), making this trade-off favorable in practice.

Exact reconstruction. To initialize the auxiliary variables model equivalently to an original model at θ_0, we set:

$$\theta_r = V_r^T\theta_0 \in \mathbb{R}^r$$
$$\theta_d = (I - V_rV_r^T)\theta_0 \in \mathbb{R}^d$$

This initialization guarantees that the reparameterized model exactly reconstructs θ_0:

$$V_r\theta_r + (I - V_rV_r^T)\theta_d = \theta_0$$

ensuring functional equivalence: $f(\theta_0) = \tilde{f}(\theta_r, \theta_d)$ for all inputs.

Algorithm D.1 EGOP Auxiliary Variables Linear Layer Forward Pass

Input: $x \in \mathbb{R}^{n \times d_{\text{in}}}$ (batch input), $\theta_r \in \mathbb{R}^r$ (reduced params), $\theta_d \in \mathbb{R}^d$ (auxiliary params), $V \in \mathbb{R}^{d \times r}$ (EGOP basis)
Output: $y \in \mathbb{R}^{n \times d_{\text{out}}}$
Layer dims: $d_{\text{out}}, d_{\text{in}}$ where $d = d_{\text{out}} \times d_{\text{in}}$
{Compute $\theta = V_r\theta_r + (I - V_rV_r^\mathsf{T})\theta_d$ using only 2 matrix multiplications}
$\theta_{\text{full}} \leftarrow \theta_d + V(\theta_r - V^\mathsf{T}\theta_d)$ \\$[d] = [d] + [d, r] \times [r]$

\\Reshape to weight matrix
$W \leftarrow \text{reshape}(\theta_{\text{full}}, (d_{\text{out}}, d_{\text{in}}))$ \\$[d_{\text{out}}, d_{\text{in}}]$

\\Apply linear transformation
$y \leftarrow xW^T + b$ \\$[n, d_{\text{out}}] = [n, d_{\text{in}}] \times [d_{\text{in}}, d_{\text{out}}]$

Return: y

D.1.2. Reparameterizing Convolutional Network Layers. To evaluate the scalability and generalization of EGOP reparameterization beyond fully connected architectures, we extend the method to convolutional neural networks (CNNs) and residual networks (ResNets). We study EGOP reparameterization on larger and deeper models such as AlexNet and ResNet trained on ImageNet, and compare performance against well tuned baseline optimizers. This extension allows us to assess whether EGOP can improve optimization in modern large scale visual recognition pipelines.

Background on convolutional layers. A convolutional layer consists of a collection of learnable filters (also called kernels). Each filter is a three-dimensional tensor of shape $k \times k \times C_{\text{in}}$, where k is the spatial size and C_{in} is the number of input channels. A layer with C_{out} output channels therefore contains C_{out} such filters, each producing one output channel by convolving the filter with the input feature map. Stacking the C_{out} resulting feature maps yields the layer's activation tensor.

Motivation for EGOP in convolutional networks. A direct application of EGOP to convolutional layers would require constructing and applying a separate reparameterization matrix for every filter, which is prohibitive for modern CNNs. However, filters within the same layer are drawn from the same initialization distribution, so their first-step gradient statistics are identically distributed; consequently, their EGOP matrices at early stages of training may also be similar. This motivates our shared reparameterization strategy: we compute a single EGOP matrix per convolutional layer and reuse it across all filters within that layer. This reduces computational cost while preserving the benefits of EGOP, enabling its use in deep architectures such as AlexNet and ResNet.

Adela: The ideas in these next two paragraph are correct, but it might be confusing for a reader to digest. Let's think about how we could make this more tractable; we could try to workshop the writing. Maybe some little graphic/block diagram could help explain waht we mean?

Shared Gradient Structure at Initialization. A key observation that enables an efficient extension of EGOP to CNNs is that, for most practical initialization distributions (e.g. Kaiming, Xavier, Glorot normal/uniform, filters within the same convolutional layer are identically distributed and so are their gradients. This property allows us to substantially reduce the computational cost of the reparameterization. During initialization, each forward/backward pass yields multiple gradients simultaneously rather than a single one, which decreases the number of required passes proportionally to the number of filters. Once these gradients are collected, we compute a single orthogonal matrix V that captures the shared structure of the layer. This V is then reused across all filters, leading to a dramatic reduction in memory footprint and computational speedup, due to the low number of required forward/backward passes required for gradient sampling.

Jose: Should I show plots here? **Adela**: Time permitting. We could include a plot comparing "shared V" versus "different V" on some small model (cifar10) to show how sharing V across filters impacts performance.

Kronecker Product Structure from Reusing V. As referenced in Section 5, reusing V across all filters as described above is equivalent to employing a Kronecker-product reparameterization matrix. Moreover, the effective Kronecker product imposed by this construction is different from that employed by SOAP/Shampoo, and the effective EGOP change-of-basis

matrix applied to each convolutional filter does *not* have any structural constraint. We clarify this point below.

In the EGOP reparameterization procedure described in the preceeding paragraphs, for each convolutional layer we form $V \in \mathbb{R}^{(k^2 C_{\text{in}}) \times (k^2 C_{\text{in}})}$ and apply this change-of-basis to each (vectorized) filter in the convolutional layer, denoted θ_j for $j = 1 \dots C_{\text{out}}$, as

$$V \operatorname{vec}(\theta_j).$$

This is equivalent to employing the following Kronecker-product structured change-of-basis

$$(\mathbb{I}_{C_{\text{out}}} \otimes V) \begin{bmatrix} \operatorname{vec}(\theta_1) \\ \vdots \\ \operatorname{vec}(\theta_{C_{\text{out}}}) \end{bmatrix}.$$

We now compare with the Kronecker-product structured change-of-basis employed by SOAP/Shampoo. For a convolutional layer with the above-described dimensions, SOAP and Shampoo form four orthogonal matrices

$$Q_h \in \mathbb{R}^{k \times k}, \quad Q_w \in \mathbb{R}^{k \times k}, \quad Q_{\text{in}} \in \mathbb{R}^{C_{\text{in}} \times C_{\text{in}}}, \quad Q_{\text{out}} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{out}}}.$$

The SOAP/Shampoo forward pass is then equivalent to performing

$$(Q_{\text{out}} \otimes Q_w \otimes Q_h \otimes Q_{\text{in}}) \begin{bmatrix} \operatorname{vec}(\theta_1) \\ \vdots \\ \operatorname{vec}(\theta_{C_{\text{out}}}) \end{bmatrix}.$$

Comparing the transformation employed in EGOP reparameterization with that employed by SOAP/Shampoo, we see that the effective transformation applied to each individual filter by SOAP/Shampoo is constrained to have a Kronecker product structure, $Q_w \otimes Q_h \otimes Q_{\text{in}}$, whereas the transformation V employed in EGOP reparameterization is a generic matrix. SOAP/Shampoo employ a more general mapping over output channels, namely Q_{out}, compared to $\mathbb{I}_{C_{\text{out}}}$ employed in EGOP reparameterization. However, as explained in the preceding paragraphs, the distribution over different filters is identical at initialization, and in practice, we find employing the identity mapping over output channels in EGOP reparameterization yields performance competitive with that of SOAP/Shampoo.

Algorithmic Implementation. The convolutional reparameterization follows the same principle as in the linear case, with additional tensor reshaping steps to accommodate convolutional filters. For each layer, we simultaneously collect gradients at initialization and flatten them into a single matrix. A full SVD is performed for each layer, and the resulting right singular vectors V define a shared orthogonal basis that captures the dominant gradient directions. Each filter is then flattened, preconditioned by V, reshaped back to its convolutional form, and used in the standard forward pass. The detailed procedure is provided below.

Algorithm D.2 EGOP Reparameterization for Convolutional Layers

Input: CNN forward map f with convolutional layers $\{W^{(l)}\}_{l=1}^{L}$, number of initializations K, sampling distribution ρ
Output: Reparameterized forward map $\tilde{f}$

\\1. Sample random initializations and collect gradients
Sample $\{\theta_k\}_{k=1}^{K} \sim \rho$ i.i.d.
for $k = 1, \dots, K$ **do**
    Initialize network parameters with θ_k
    Perform one forward/backward pass
    Store gradients $\{\nabla W^{(l,k)}\}_{l=1}^{L}$
end for

\\2. Form per-layer gradient matrices
for each convolutional layer $l = 1, \dots, L$ **do**
    \\Flatten each kernel gradient $\nabla W_j^{(l,k)} \in \mathbb{R}^{c_{\text{in}} \times k_h \times k_w}$ into a vector
    \\Stack across initializations k and output channels j
    $G^{(l)} \leftarrow \text{stack}\Big(\text{flatten}(\nabla W_j^{(l,k)})\Big)_{k=1,\dots,K;\, j=1,\dots,c_{\text{out}}}$ \\$G^{(l)} \in \mathbb{R}^{(c_{\text{in}} k_h k_w) \times (K c_{\text{out}})}$
end for

\\3. Compute EGOP eigenbasis per layer
for each layer $l = 1, \dots, L$ **do**
    $V^{(l)} \leftarrow$ `left_singular_vectors`$(G^{(l)})$
end for

\\4. Define reparameterized forward map
Define $\tilde{f}(\{W_j^{(l)}\}_{j=1}^{L}) = f\left(\left\{\text{reshape}\Big(V^{(l)}\, \text{flatten}(W_j^{(l)}), (c_{\text{in}}, k_h, k_w)\Big)\right\}_{j=1}^{L}\right)$

\\5. Optimize the reparameterized model
Optimize $\tilde{f}$ using an adaptive optimizer of choice
return $\tilde{f}$

Appendix E. Deferred Proofs.

E.1. Deferred Proofs from Section A.1. In this section, we will leverage the following basic results from linear algebra, whose proofs we include for completeness.

Lemma E.1. *Given $A \in \mathbb{R}^{d \times d}$ a symmetric PSD matrix, for any vector $v \in \mathbb{R}^d$,*

$$\langle v, A^{\mathsf{T}} A v \rangle \leq \lambda_{\max}(A) \langle v, A v \rangle$$

where $\lambda_{\max}(A)$ denotes the largest eigenvalue of A.

Proof of Lemma E.1. Since A is PSD, it admits a square root $A^{1/2}$ which is itself PSD. Consequently,

$$\langle Av, Av\rangle = \|Av\|^2 \le \|A^{1/2}\|_{\text{op}}^2 \|A^{1/2}v\|^2 = \lambda_{\max}(A)\langle v, Av\rangle. \qquad \blacksquare$$

Lemma E.2 (Order-preserving square root). *For any positive-semidefinite matrices X, Y, the following holds:*

$$X \preceq Y \implies X^{1/2} \preceq Y^{1/2}. \tag{E.1}$$

Proof of Lemma A.7. Because $\tilde{\rho}(\tilde{\theta}) = \rho(V\tilde{\theta})$,

$$\mathbb{E}_{\tilde{\theta}\sim\tilde{\rho}}[q(\tilde{\theta})] = \mathbb{E}_{\theta\sim\rho}[q(V^\mathsf{T}\theta)]$$

By the chain rule,

$$\nabla\tilde{f}(\tilde{\theta}) = V^\mathsf{T}\nabla f(V\tilde{\theta}).$$

Thus

$$\mathbb{E}_{\tilde{\theta}\sim\tilde{\rho}}[\nabla\tilde{f}(\tilde{\theta})\nabla\tilde{f}(\tilde{\theta})^\mathsf{T}] = V^\mathsf{T}\mathbb{E}_{\tilde{\theta}\sim\tilde{\rho}}[\nabla f(V\tilde{\theta})f(V\tilde{\theta})^\mathsf{T}]V = V^\mathsf{T}\mathbb{E}_{\theta\sim\rho}[\nabla f(\theta)\nabla f(\theta)^\mathsf{T}]V = \Lambda. \qquad \blacksquare$$

Proof of Lemma E.2. Since $Y - X \succeq 0$, any unit vector v satisfies

$$0 \le \langle v, (Y - X)v\rangle = \langle v, (Y^{1/2} + X^{1/2})(Y^{1/2} - X^{1/2})v\rangle.$$

Since X, Y are PSD, their square roots are well-defined and symmetric. In particular, the matrix $Y^{1/2} - X^{1/2}$ is symmetric and, as such, has real eigenvalues. Therefore, it suffices to prove all eigenvalues of this matrix are nonnegative.

To that end, let (λ, v) be an eigenpair of $Y^{1/2} - X^{1/2}$. We have

$$0 \le \langle (Y^{1/2} - X^{1/2})v, (Y^{1/2} + X^{1/2})v\rangle = \lambda\langle v, (Y^{1/2} + X^{1/2})v\rangle. \tag{E.2}$$

In particular, assume $\lambda < 0$ (since otherwise there is nothing to show). Consequently,

$$0 > \lambda = \langle v, (Y^{1/2} - X^{1/2})v\rangle \implies \langle v, X^{1/2}v\rangle > \langle v, Y^{1/2}v\rangle \ge 0,$$

where the last inequality follows from the fact that Y is PSD. In particular, this implies that $\langle v, (Y^{1/2} + X^{1/2})v\rangle > 0$; combined with (E.2) and the assumption $\lambda < 0$, this yields a contradiction. Therefore, $\lambda \ge 0$. Since the choice of eigenpair was arbitrary, the claim follows. $\blacksquare$

Given these lemmas, we are now equipped to present full proofs of the results presented in Section A.1.

Proof of Lemma A.8. An intermediate result in the proof of Thm. 4.1 in Liu et al. [25] implies that in the case of noise-free gradients, i.e. oracle gradient $g(\cdot) = \nabla\tilde{f}(\cdot)$, Adagrad with constraint set $\tilde{\Theta}$ produces iterates such that

$$\frac{1}{T}\sum_{t=0}^{T-1}(\tilde{f}(\tilde{\theta}_t) - \tilde{f}(\theta^*)) \le \left(\eta + \frac{\tilde{D}_\infty^2}{\eta}\right)^2 \cdot \frac{\|\vec{\tilde{L}}\|_1}{T} + \frac{\epsilon\tilde{D}_2^2}{\eta T}.$$

We note $D_2^2 = \tilde{D}_2^2$ because the Euclidean norm is invariant under orthonormal transformation. Because the constraint update implies $\{\tilde{\theta}_t\} \subseteq \tilde{\Theta}$, applying Thm. A.1 implies

$$\|\vec{\tilde{L}}\|_1 \leq \|\vec{L}\|_1 \cdot \sqrt{1+\delta}\left(\frac{2\,\mathrm{sr}(f)}{d(\beta_1-\delta)} + 2\frac{\sqrt{\delta}}{\beta_1-\delta}\left(1+\frac{2}{\sqrt{d}}\right)\right)$$

which immediately implies the result. ∎

Proof of Lemma A.5. By definition, f is twice-differentiable if and only if it satisfies

$$\text{(E.3)}\quad f(y) = f(x) + \langle \nabla f(x), y-x\rangle + \langle y-x, \nabla^2 f(x)(y-x)\rangle + o(\|y-x\|^2), \quad \text{for all } x, y \in \mathbb{R}^d.$$

We first prove the "if" version: assume $|\langle v, \nabla^2 f(\theta) v\rangle| \leq \langle v, \mathrm{diag}(L) v\rangle$. Consider any $x, y \in \Theta$. By the mean value theorem,

$$\begin{aligned} f(y) - f(x) - \langle \nabla f(x), y-x\rangle &= \int_0^1 t\langle y-x, \nabla^2 f(x+t(y-x))(y-x)\rangle \mathrm{d}t \\ &\leq \int_0^1 t\langle y-x, \mathrm{diag}(L)(y-x)\rangle \mathrm{d}t \\ &= \langle y-x, \mathrm{diag}(L)(y-x)\rangle, \end{aligned}$$

where the inequality follows by assumption and the convexity of Θ. Taking absolute values on both sides yields the claim.

We now prove the "only if" part. In particular, for any $x, y \in \Theta$, writing $y = x + tv$ and invoking (E.3) we obtain

$$\begin{aligned} |\langle y-x, \nabla^2 f(x)(y-x)\rangle| &\leq |f(y) - f(x) - \langle \nabla f(x), y-x\rangle| + o(\|y-x\|^2) \\ &\leq \langle y-x, \mathrm{diag}(L)(y-x)\rangle + o(\|y-x\|^2) \\ \Leftrightarrow\ t^2|\langle v, \nabla^2 f(x) v\rangle| &\leq t^2\langle v, \mathrm{diag}(L) v\rangle + o(t^2). \end{aligned}$$

∎

Because t was chosen such that $y = x + tv$ for $x, y \in \Theta$, convexity of Θ implies that $x + \tau v \in \Theta$ for all $\tau \in [0, t]$. Thus dividing both sides by t and letting $t \downarrow 0$ implies the result.

Proof of Lemma A.6. Given $f(\cdot)$ satisfying Assumption 2, $\forall \theta_1, \theta_2 \in \Theta$

$$\|\nabla f(\theta_1) - \nabla f(\theta_2) - \nabla^2 f(\theta_2)(\theta_1 - \theta_2)\|_2 \leq H\|\theta_1 - \theta_2\|_2^2.$$

For fixed θ^* the local minimum in Assumption 1, this implies that $\forall \theta \in \Theta\ \exists R_f(\theta) \in \mathbb{R}^d$ such that

$$\nabla f(\theta) = \nabla f(\theta^*) + \nabla^2 f(\theta^*)(\theta - \theta^*) + R_f(\theta) \quad \text{and} \quad \|R_f(\theta)\|_2 \leq H\|\theta - \theta^*\|_2^2.$$

Given θ^* a stationary point of $f(\cdot)$, $\nabla f(\theta^*) = 0$, so the above simplifies to

$$\nabla f(\theta) = \nabla^2 f(\theta^*)(\theta - \theta^*) + R_f(\theta).$$

We can substitute the above expression into the definition of the EGOP matrix:

$$\begin{aligned}\mathbb{E}_{\theta\sim\rho}[\nabla f(\theta)\nabla f(\theta)^\mathsf{T}] &= \mathbb{E}_{\theta\sim\rho}\left[\left(\nabla^2 f(\theta^*)(\theta-\theta^*)+R_f(\theta)\right)\left(\nabla^2 f(\theta^*)(\theta-\theta^*)+R_f(\theta)\right)^\mathsf{T}\right]\\ &= \mathbb{E}_{\theta\sim\rho}\left[\nabla^2 f(\theta^*)(\theta-\theta^*)(\theta-\theta^*)^\mathsf{T}\nabla^2 f(\theta^*)^\mathsf{T}\right]+E_f(\theta^*)\end{aligned}$$

where

$$E_f(\theta^*) \stackrel{\text{def}}{=} \mathbb{E}_{\theta\sim\rho}[R_f(\theta)(\theta-\theta^*)^\mathsf{T}\nabla^2 f(\theta^*)^\mathsf{T}+\nabla^2 f(\theta^*)(\theta-\theta^*)R_f(\theta)^\mathsf{T}+R_f(\theta)R_f(\theta)^\mathsf{T}].$$

We can simplify the first term by leveraging the fact that ρ is isotropic about θ_c, as specified in Assumption 1:

$$\begin{aligned}\mathbb{E}_{\theta\sim\rho}\left[\nabla^2 f(\theta^*)(\theta-\theta^*)(\theta-\theta^*)^\mathsf{T}\nabla^2 f(\theta^*)^\mathsf{T}\right] &= \nabla^2 f(\theta^*)\mathbb{E}_{\theta\sim\rho}\left[(\theta-\theta^*)(\theta-\theta^*)^\mathsf{T}\right]\nabla^2 f(\theta^*)^\mathsf{T}\\ &= \nabla^2 f(\theta^*)\mathbb{E}_{\theta\sim\rho}\left[(\theta-\theta_c+(\theta_c-\theta^*))(\theta-\theta_c+(\theta_c-\theta^*))^\mathsf{T}\right]\nabla^2 f(\theta^*)^\mathsf{T}\\ &= \nabla^2 f(\theta^*)\left(\mathbb{E}_{\theta\sim\rho}\left[(\theta_c-\theta)(\theta_c-\theta)^\mathsf{T}\right]+(\theta_c-\theta^*)(\theta_c-\theta^*)^\mathsf{T}\right)\nabla^2 f(\theta^*)^\mathsf{T}\\ &= \nabla^2 f(\theta^*)\left(c^2\mathbb{I}+(\theta_c-\theta^*)(\theta_c-\theta^*)^\mathsf{T}\right)\nabla^2 f(\theta^*)^\mathsf{T}\end{aligned}$$

where the penultimate equality follows from the assumption that ρ has mean θ_c and the last equality follows from the assumption that ρ is isotropic about θ_c. Because θ^* is a local minimum, $\nabla^2 f(\theta^*)$ is PSD, and further the rank-1 outer product is also PSD, so we have

$$\nabla^2 f(\theta^*)\left(c^2\mathbb{I}+(\theta_c-\theta^*)(\theta_c-\theta^*)^\mathsf{T}\right)\nabla^2 f(\theta^*)^\mathsf{T} \succeq c^2\nabla^2 f(\theta^*)\nabla^2 f(\theta^*)^\mathsf{T}.$$

Moreover by Assumption 1, $\|\theta_c-\theta^*\|_2^2 \le c^2$, and thus

$$(\theta_c-\theta^*)(\theta_c-\theta^*)^\mathsf{T} \preceq c^2\mathbb{I}.$$

This implies

$$\nabla^2 f(\theta^*)\left(c^2\mathbb{I}+(\theta_c-\theta^*)(\theta_c-\theta^*)^\mathsf{T}\right)\nabla^2 f(\theta^*)^\mathsf{T} \preceq 2c^2\nabla^2 f(\theta^*)\nabla^2 f(\theta^*)^\mathsf{T}.$$

Combining these two matrix inequalities yields the matrix inequalities for

$$G(\theta^*) \stackrel{\text{def}}{=} \nabla^2 f(\theta^*)\left(c^2\mathbb{I}+(\theta_c-\theta^*)(\theta_c-\theta^*)^\mathsf{T}\right)\nabla^2 f(\theta^*)^\mathsf{T}.$$

Lastly, we bound $|\langle v, E_f(\theta^*)v\rangle|$ for $E_f(\theta^*)$ defined above. Applying triangle inequality, Jensen's inequality, and Cauchy-Schwarz, to the definition of $E_f(\theta^*)$ yields

$$\begin{aligned}|\langle v, E_f(\theta^*)v\rangle| &\le \mathbb{E}_{\theta\sim\rho}\left[2|\langle v, R_f(\theta)\rangle|\cdot|\langle\theta-\theta^*, \nabla^2 f(\theta^*)v\rangle| + |\langle v, R_f(\theta)\rangle^2|\right]\\ &\le \mathbb{E}_{\theta\sim\rho}\left[2\|R_f(\theta)\|_2\|\theta-\theta^*\|_2\lambda_{\max}(\nabla^2 f(\theta^*))\|v\|_2^2 + \|R_f(\theta)\|_2^2\|v\|_2^2\right].\end{aligned}$$

Recall that under Assumption 2, $\|R_f(\theta)\|_2 \le H\|\theta-\theta^*\|_2^2$. Substituting this into the above bound yields

$$|\langle v, E_f(\theta^*)v\rangle| \le \left(2H\lambda_{\max}(\nabla^2 f(\theta^*))M_3 + H^2 M_4\right)\|v\|_2^2$$

where

$$M_p \stackrel{\text{def}}{=} \mathbb{E}_{\theta\sim\rho}[\|\theta-\theta^*\|_2^p].$$

■

Proof of Lemma A.3. By Lemma A.5, the smoothness constants $L_1, \dots, L_d$ must satisfy

$$\forall \theta \in \Theta, \forall v, \quad \langle v, \operatorname{diag}(L) v \rangle \geq |\langle v, \nabla^2 f(\theta) v \rangle|.$$

where $\operatorname{diag}(L) \in \mathbb{R}^{d \times d}$ is the diagonal matrix with entries $\operatorname{diag}(L)_{i,i} = L_i$. In particular, consider $\theta = \theta^*$, and let $\nu \in \{\pm d^{-1/2}\}^d$ be a dense unit vector whose entries all satisfy $|\nu(i)| = d^{-1/2}$. Then the smoothness constants must satisfy

$$\langle \nu, \operatorname{diag}(L) \nu \rangle = \frac{1}{d} \sum_{i=1}^{d} L_i \geq \langle \nu, \nabla^2 f(\theta^*) \nu \rangle.$$

Given θ^* a local minimum, $\nabla^2 f(\theta^*)$ is PSD. Thus by Lemma E.1,

$$\langle \nu, \nabla^2 f(\theta^*) \nu \rangle \geq \frac{1}{\lambda_{\max}(\nabla^2 f(\theta^*))} \langle \nu, \nabla^2 f(\theta^*) \nabla^2 f(\theta^*)^{\mathsf{T}} \nu \rangle$$

where $\lambda_{\max}(\nabla^2 f(\theta^*))$ denotes the leading eigenvalue of $\nabla^2 f(\theta^*)$. Moreover, Lemma A.6 implies

$$\langle \nu, \nabla^2 f(\theta^*) \nabla f(\theta^*)^{\mathsf{T}} \nu \rangle \geq \frac{1}{2c^2} \langle \nu, G(\theta^*) \nu \rangle = \frac{1}{2c^2} \langle \nu, \left(\mathbb{E}_{\theta \sim \rho}[\nabla f(\theta) \nabla f(\theta)^{\mathsf{T}}] - E_f(\theta^*) \right) \nu \rangle.$$

$$\langle v_1, \nabla^2 f(\theta^*) \nabla f(\theta^*)^{\mathsf{T}} v_1 \rangle \geq \frac{\lambda_{\max}(\mathrm{EGOP}) - \langle v_1, E_f(\theta^*) v_1 \rangle}{2c^2}.$$

Chaining the preceding inequalities yields

$$\langle \nu, \operatorname{diag}(L) \nu \rangle \geq \frac{\langle \nu, \mathrm{EGOP}\, \nu \rangle - \langle \nu, E_f(\theta^*) \nu \rangle}{2c^2 \lambda_{\max}(\nabla^2 f(\theta^*))}. \tag{E.4}$$

Given $\nabla^2 f(\theta^*)$ PSD, we have

$$\lambda_{\max}(\nabla^2 f(\theta^*)) = \sqrt{\lambda_{\max}(\nabla^2 f(\theta^*))^2} = \sqrt{\lambda_{\max}(\nabla^2 f(\theta^*) \nabla^2 f(\theta^*)^{\mathsf{T}})}.$$

Moreover, by Lemma A.6 we have the following upper bound:

$$\begin{aligned}
\lambda_{\max}(\nabla^2 f(\theta^*) \nabla^2 f(\theta^*)^{\mathsf{T}}) &\leq \frac{1}{c^2} \lambda_{\max}(G(\theta^*)) \\
&= \frac{1}{c^2} \max_{v \in \mathbb{R}^d} \langle v, (\mathbb{E}_{\theta \sim \rho}[\nabla f(\theta) \nabla f(\theta)^{\mathsf{T}}] - E_f(\theta^*)) v \rangle \\
&\leq \frac{1}{c^2} \left(\lambda_{\max}(\mathrm{EGOP}) + \lambda_{\max}(E_f(\theta^*)) \right).
\end{aligned}$$

Combining this inequality with the preceding equation implies

$$\lambda_{\max}(\nabla^2 f(\theta^*)) \leq \frac{1}{c} \sqrt{\lambda_{\max}(\mathrm{EGOP}) + \lambda_{\max}(E_f(\theta^*))}.$$

Substituting this inequality into Eq. E.4 implies

$$\langle \nu, \mathrm{diag}(L)\nu \rangle \geq \frac{\langle \nu, \mathrm{EGOP}\, \nu \rangle - \langle \nu, E_f(\theta^*)\nu \rangle}{2c\sqrt{\lambda_{\max}(\mathrm{EGOP}) + \lambda_{\max}(E_f(\theta^*))}}.$$

Recall that $\nu \in \{\pm d^{-1/2}\}^d$. Thus

$$\langle \nu, \mathrm{diag}(L)\nu \rangle = \sum_{i=1}^{d} L_i v(i)^2 = \frac{1}{d}\sum_{i=1}^{d} L_i.$$

Combining this with the above inequality implies

$$\sum_{i=1}^{d} L_i \geq \frac{d\left(\langle \nu, \mathrm{EGOP}\, \nu \rangle - \langle \nu, E_f(\theta^*)\nu \rangle\right)}{2c\sqrt{\lambda_{\max}(\mathrm{EGOP}) + \lambda_{\max}(E_f(\theta^*))}}.$$

By Lemma A.6, $\forall v \in \mathbb{R}^d$

$$|\langle v, E_f(\theta^*)v \rangle| \leq \gamma \|v\|_2^2$$

where

$$\gamma \stackrel{\mathrm{def}}{=} 2H\lambda_{\max}(\nabla^2 f(\theta^*))M_3 + H^2 M_4$$

Thus

$$\sum_{i=1}^{d} L_i \geq \frac{d}{2c}\frac{\langle \nu, \mathrm{EGOP}\, \nu \rangle - \gamma}{\sqrt{\lambda_{\max}(\mathrm{EGOP}) + \gamma}}.$$

As this holds for all $\nu \in \{\pm d^{-1/2}\}^d$, we conclude

$$\sum_{i=1}^{d} L_i \geq \frac{d}{2c} \cdot \max_{\nu \in \{\pm d^{-1/2}\}^d} \frac{\langle \nu, \mathrm{EGOP}\, \nu \rangle - \gamma}{\sqrt{\lambda_{\max}(\mathrm{EGOP}) + \gamma}}.$$

We observe that for any $v \in \mathbb{R}^d$,

$$\langle v, \mathrm{EGOP}\, v \rangle = \sum_{i=1}^{d} \lambda_i \langle v, v_i \rangle^2 \geq \max_k \lambda_k \langle v, v_k \rangle^2$$

where (λ_i, v_i) denote the i^{th} eigenvalue and eigenvector of the EGOP indexed in decreasing order of magnitude. In particular, for any v_k, consider $\nu_k \in \mathbb{R}^d$ defined to have entries $\nu_k(i) = \mathrm{sign}(v_k(i))d^{-1/2}$. Observe that

$$\nu_k \in \mathrm{argmax}\{\langle \nu, v_k \rangle^2 \mid \nu \in \{\pm d^{-1/2}\}^d\}.$$

For this ν_k,

$$\langle \nu, v_k \rangle^2 = \left(\sum_{i=1}^{d} \mathrm{sign}(v_k(i))d^{-1/2}v_k(i)\right)^2 = \frac{1}{d}\left(\sum_{i=1}^{d}|v_k(i)|\right)^2 = \frac{1}{d}\|v_k\|_1^2.$$

Thus in general,

$$\max_{\nu\in\{\pm d^{-1/2}\}^d} \langle \nu, \mathrm{EGOP}\,\nu\rangle = \max_{\nu\in\{\pm d^{-1/2}\}^d} \sum_{k=1}^{d} \lambda_k \langle \nu, v_k\rangle^2 \geq \max_k \frac{\lambda_k}{d}\|v_k\|_1^2.$$

This implies the lower bound

$$\sum_{i=1}^{d} L_i \geq \frac{d}{2c}\cdot \max_k \frac{\lambda_k \|v_k\|_1^2/d - \gamma}{\sqrt{\lambda_{\max}(\mathrm{EGOP}) + \gamma}}.$$

■

Proof of Lemma A.4. Consider reparameterized function $\tilde{f}(\theta) \stackrel{\text{def}}{=} f(V\theta)$ and corresponding $\tilde{\Theta}$ and $\tilde{\rho}$ as defined in Eq. A.3. Observe that Assumption 2 implies $\nabla^2 f(\cdot)$ is also H-Lipschitz within $\tilde{\Theta}$ and that $V^T\theta^* \in \tilde{\Theta}$ is a local minimum. Applying Lemma A.6 to $\tilde{f}(\cdot)$ implies

$$\mathbb{E}_{\tilde{\rho}}[\nabla\tilde{f}(\theta)\nabla\tilde{f}(\theta)^T] = G(V^T\theta^*) + E_{\tilde{f}}[V^T\theta^*]$$

where

$$c^2\nabla^2\tilde{f}(V^T\theta^*)\nabla^2\tilde{f}(V^T\theta^*)^T \preceq G(V^T\theta^*) \preceq 2c^2\nabla^2\tilde{f}(V^T\theta^*)\nabla^2\tilde{f}(V^T\theta^*)^T$$

and

$$|\langle v, E_{\tilde{f}}[V^T\theta^*]v\rangle| \leq \gamma\|v\|_2^2$$

for

$$\gamma \stackrel{\text{def}}{=} 2H\lambda_{\max}(\nabla^2 f(\theta^*))M_3 + H^2 M_4.$$

We've used the fact that

$$M_p \stackrel{\text{def}}{=} \mathbb{E}_{\theta\sim\rho}[\|\theta-\theta^*\|_2^p] = \mathbb{E}_{\theta\sim\rho}[\|V^\mathsf{T}(\theta-\theta^*)\|_2^p] = \mathbb{E}_{\tilde{\theta}\sim\tilde{\rho}}[\|\tilde{\theta}-V^\mathsf{T}\theta^*\|_2^p]$$

and

$$\lambda_{\max}(\nabla^2\tilde{f}(V^\mathsf{T}\theta^*))$$

to bound $|\langle v, E_{\tilde{f}}[V^T\theta^*]v\rangle|$ by the same value γ.

Combining the above matrix inequalities with the implication $E_{\tilde{f}}[V^T\theta^*] \preceq \gamma\mathbb{I}$ implies

$$c^2\nabla^2\tilde{f}(V^\mathsf{T}\theta^*)\nabla^2\tilde{f}(V^\mathsf{T}\theta^*)^\mathsf{T} \preceq \mathbb{E}_{\theta\sim\rho}[\nabla\tilde{f}(\theta)\nabla\tilde{f}(\theta)^\mathsf{T}] + \gamma\mathbb{I}$$

Recall that the PSD ordering $X \preceq Y$ implies $X^{1/2} \preceq Y^{1/2}$ (for completeness, see statement and proof in Lemma E.2). Thus the preceding matrix inequality implies

$$\nabla^2\tilde{f}(V^\mathsf{T}\theta^*) \preceq \frac{1}{c}\left(\mathbb{E}_{\theta\sim\rho}[\nabla\tilde{f}(\theta)\nabla\tilde{f}(\theta)^\mathsf{T}] + \gamma\mathbb{I}\right)^{1/2}$$

Lemma A.7 implies $\mathbb{E}_{\tilde{\rho}}[\nabla\tilde{f}(\theta)\nabla\tilde{f}(\theta)^\mathsf{T}]$ is diagonal, and its entries are given by the eigenvalues of $\mathrm{EGOP}(f) \stackrel{\text{def}}{=} \mathbb{E}_{\theta\sim\rho}[\nabla f(\theta)\nabla f(\theta)^\mathsf{T}]$. As the EGOP is an expectation of PSD matrices, these eigenvalues are all non-negative, and thus

$$\mathbb{E}_{\theta\sim\tilde{\rho}}[\nabla\tilde{f}(\theta)\nabla\tilde{f}(\theta)^\mathsf{T}]_{i,i}^{1/2} = \sqrt{\lambda_i(\mathrm{EGOP}(f))}.$$

Since $\mathbb{E}_{\theta\sim\rho}[\nabla\tilde{f}(\theta)\nabla\tilde{f}(\theta)^\mathsf{T}]$ is diagonal and $\gamma\mathbb{I}$ is diagonal,

$$\left(\mathbb{E}_{\theta\sim\rho}[\nabla\tilde{f}(\theta)\nabla\tilde{f}(\theta)^\mathsf{T}]+\gamma\mathbb{I}\right)^{1/2}_{i,i}=\sqrt{\mathbb{E}_{\theta\sim\rho}[\nabla\tilde{f}(\theta)\nabla\tilde{f}(\theta)^\mathsf{T}]_{i,i}+\gamma}.$$

Thus for all $v\in\mathbb{R}^d$, we have

$$\langle v,\nabla^2\tilde{f}(V^\mathsf{T}\theta^*)v\rangle\leq\frac{1}{c}\sum_{i=1}^d v(i)^2\sqrt{\mathbb{E}_{\theta\sim\rho}[\nabla\tilde{f}(\theta)\nabla\tilde{f}(\theta)^\mathsf{T}]_{i,i}+\gamma}. \tag{E.5}$$

Employing reverse triangle inequality and the fact that $V^T\theta^*$ is a local minimum implies $\nabla^2\tilde{f}(V^T\theta^*)$ is PSD, we can show that Assumption 2 implies

$$|\langle v,\nabla^2\tilde{f}(\theta)v\rangle|\leq\langle v,\nabla^2\tilde{f}(V^T\theta^*)v\rangle+H\|\theta-V^T\theta^*\|_2\|v\|_2^2$$

that $\forall\theta\in\tilde{\Theta},\forall v\in\mathbb{R}^d$. Observe that $\|\theta-V^T\theta^*\|_2\leq\tilde{D}_2=D_2$ for D_2 the ℓ_2-diameter of Θ. Thus combining the above with Eq. E.5 implies $\forall\theta\in\tilde{\Theta},\forall v\in\mathbb{R}^d$,

$$|\langle v,\nabla^2\tilde{f}(\theta)v\rangle|\leq\sum_{i=1}^d\sqrt{\lambda_i(\mathrm{EGOP}(f)+\gamma}/c+HD_2)v(i)^2.$$

Thus by Lemma A.5, this implies that $\tilde{f}(\cdot)$ satisfies coordinate-wise smoothness with respect to constants

$$L_i\leq\frac{1}{c}\sqrt{\lambda_i(\mathrm{EGOP}(f))+\gamma}+HD_2.$$

This implies the sum of the coordinate-wise smoothness constants is bounded above as

$$\begin{aligned}L_{\tilde{f}}&\leq\sum_{i=1}^d\left(\frac{1}{c}\sqrt{\lambda_i(\mathrm{EGOP}(f))+\gamma}+HD_2\right)\\&\leq d\left(\frac{\sqrt{\gamma}}{c}+HD_2\right)+\frac{1}{c}\sum_{i=1}^d\sqrt{\lambda_i(\mathrm{EGOP}(f))}.\end{aligned}$$

■

E.1.1. Lipschitz Constants of Hessians in Machine Learning. In this section, we note one family of naturally-motivated non-convex objectives with locally Lipschitz Hessians that arise in machine learning problems. We consider the over-parameterized matrix factorization problem: let $\theta\in\mathbb{R}^{(d_1+d_2)k}$ be parameters, whose entries can be grouped into two matrices as $\theta=(L,R)$ for $L\in\mathbb{R}^{d_1\times k},R\in\mathbb{R}^{d_2\times k}$. We define the objective

$$f(\theta)=\|\mathcal{A}(LR^\mathsf{T})-b\|_2^2 \tag{E.6}$$

where $b\in\mathbb{R}^m$ correspond to measurements of some matrix under map $\mathcal{A}(\cdot)$, and where $\mathcal{A}:\mathbb{R}^{d_1\times d_2}\to\mathbb{R}^m$ denotes a map

$$\mathcal{A}(X)\stackrel{\mathrm{def}}{=}(\langle A_1,X\rangle,\ldots,\langle A_m,X\rangle)\in\mathbb{R}^m.$$

We note that one a special case of Eq. E.6 is the set of objectives that arise when training a two-layer linear feed-forward network using mean-squared-error loss.

We can bound the Lipschitz constant of the Hessian of this objectives in this family:

Lemma E.3. *Consider the overparameterized matrix factorization objective with a linear measurement map, as defined in Eq. E.6. Let $\theta \in \mathbb{R}^{(d_1+d_2)k}$ be parameters, whose entries can be grouped into two matrices: $\theta = (L, R)$ for $L \in \mathbb{R}^{d_1 \times k}, R \in \mathbb{R}^{d_2 \times k}$. Consider any measurement vector $b \in \mathbb{R}^m$. Then in the ball $\|\theta\|_2 \le B$, the objective in Eq. E.6 satisfies*

$$\|\nabla^2 f(\theta_1) - \nabla^2 f(\theta_2)\|_{\text{op}} \le 12B \left(\sum_{i=1}^{m} \|A_i\|_{\mathsf{F}}^2 \right) \|\theta_1 - \theta_2\|_2.$$

To prove this result, we first characterize the quadratic form of the Hessian of Eq. E.6.

Lemma E.4. *For the objective $f(\cdot)$ as defined in Eq. E.6, for any $\theta, v \in \mathbb{R}^{(d_1 \times d_2)k)}$ with entries denoted $\theta = (L, R)$ and $v = (U, V)$, the Hessian quadratic form can be expressed as*

$$D^2 f(L, R)[U, V] = 2\|\mathcal{A}(UR^{\mathsf{T}} + LV^{\mathsf{T}})\|_2^2 + 4\langle \mathcal{A}(LR^{\mathsf{T}}) - b, \mathcal{A}(UV^{\mathsf{T}})\rangle.$$

Proof. We begin by deriving the gradient form using the limit definition:

$$\begin{aligned}
\nabla f(L, R)[U, V] &\stackrel{\text{def}}{=} \lim_{t \to 0} \frac{1}{t} \left(f(L + tU, R + tV) - f(L, R) \right) \\
&= \lim_{t \to 0} \frac{1}{t} \left(\|\mathcal{A}((L + tU)(R + tV)^{\mathsf{T}}) - b\|_2^2 - \|\mathcal{A}(LR^{\mathsf{T}}) - b\|_2^2 \right).
\end{aligned}$$

By linearity of the map $\mathcal{A}(\cdot)$,

$$\begin{aligned}
\mathcal{A}((L + tU)(R + tV)^{\mathsf{T}}) &= \mathcal{A}(LR^{\mathsf{T}} + tUR^{\mathsf{T}} + tLV^{\mathsf{T}} + t^2 UV^{\mathsf{T}}) \\
&= \mathcal{A}(LR^{\mathsf{T}}) + t\mathcal{A}(UR^{\mathsf{T}} + LV^{\mathsf{T}}) + t^2 \mathcal{A}(UV^{\mathsf{T}}).
\end{aligned}$$

Substituting this into the above limit yields

$$\nabla f(L, R)[U, V] = \lim_{t \to 0} \frac{1}{t} \left(\|\Delta + t\mathcal{A}(UR^{\mathsf{T}} + LV^{\mathsf{T}}) + t^2 \mathcal{A}(UV^{\mathsf{T}})\|_2^2 - \|\Delta\|_2^2 \right)$$

where $\Delta \stackrel{\text{def}}{=} \mathcal{A}(LR^{\mathsf{T}}) - b$. Expanding the squared norm, this yields

$$\begin{aligned}
\nabla f(L, R)[U, V] &= \lim_{t \to 0} \frac{1}{t} \Big(\|\Delta\|_2^2 + \|t\mathcal{A}(UR^{\mathsf{T}} + LV^{\mathsf{T}}) + t^2 \mathcal{A}(UV^{\mathsf{T}})\|_2^2 \\
&\qquad + 2\langle \Delta, t\mathcal{A}(UR^{\mathsf{T}} + LV^{\mathsf{T}}) + t^2 \mathcal{A}(UV^{\mathsf{T}})\rangle - \|\Delta\|_2^2 \Big) \\
&= \lim_{t \to 0} \frac{1}{t} \Big(\|t\mathcal{A}(UR^{\mathsf{T}} + LV^{\mathsf{T}}) + t^2 \mathcal{A}(UV^{\mathsf{T}})\|_2^2 \\
&\qquad + 2\langle \Delta, t\mathcal{A}(UR^{\mathsf{T}} + LV^{\mathsf{T}}) + t^2 \mathcal{A}(UV^{\mathsf{T}})\rangle \Big) \\
&= \lim_{t \to 0} \Big(t\|\mathcal{A}(UR^{\mathsf{T}} + LV^{\mathsf{T}}) + t\mathcal{A}(UV^{\mathsf{T}})\|_2^2 \\
&\qquad + 2\langle \Delta, \mathcal{A}(UR^{\mathsf{T}} + LV^{\mathsf{T}}) + t\mathcal{A}(UV^{\mathsf{T}})\rangle \Big) \\
&= 0 + 2\langle \Delta, \mathcal{A}(UR^{\mathsf{T}} + LV^{\mathsf{T}}) + 0\rangle \\
&= 2\langle \mathcal{A}(LR^{\mathsf{T}}) - b), \mathcal{A}(UR^{\mathsf{T}} + LV^{\mathsf{T}}\rangle
\end{aligned}$$

where the last line follows from $\Delta \stackrel{\text{def}}{=} \mathcal{A}(LR^\mathsf{T}) - b$. Given this expression for the gradient, we can then define the Hessian quadratic form on input $v = (U, V)$ as

$$\nabla^2 f(L,R)[U,V] \stackrel{\text{def}}{=} \lim_{t\to 0} \frac{1}{t}\left(\nabla f(L+tU, R+tV)[U,V] - \nabla f(L,R)[U,V]\right).$$

Given the above expression for the gradient, this yields

$$\nabla^2 f(L,R)[U,V] = \lim_{t\to 0} \frac{1}{t}\Big(2\langle \mathcal{A}((L+tU)(R+tV)^\mathsf{T}) - b, \mathcal{A}(U(R+tV)^\mathsf{T} + (L+tU)V^\mathsf{T})\rangle - 2\langle \Delta, \mathcal{A}(UR^\mathsf{T} + LV^\mathsf{T})\Big).$$

As noted above,

$$\mathcal{A}((L+tU)(R+tV)^\mathsf{T}) = \mathcal{A}(LR^\mathsf{T}) + t\mathcal{A}(UR^\mathsf{T} + LV^\mathsf{T}) + t^2\mathcal{A}(UV^\mathsf{T}).$$

Substituting this in to the first term in the expression for the Hessian, and using linearity of $\mathcal{A}(\cdot)$ to simplify terms, yields

$$\begin{aligned}
\nabla^2 f(L,R)[U,V] &= \lim_{t\to 0} \frac{2}{t}\Big(\langle \Delta + t\mathcal{A}(UR^\mathsf{T} + LV^\mathsf{T}) + t^2\mathcal{A}(UV^\mathsf{T}), \mathcal{A}(UR^\mathsf{T} + LV^\mathsf{T}) + 2t\mathcal{A}(UV^\mathsf{T})\rangle \\
&\qquad - \langle \Delta, \mathcal{A}(UR^\mathsf{T} + LV^\mathsf{T})\Big) \\
&= \lim_{t\to 0} \frac{2}{t}\Big(\langle \Delta, \mathcal{A}(UR^\mathsf{T} + LV^\mathsf{T})\rangle + \langle t\mathcal{A}(UR^\mathsf{T} + LV^\mathsf{T}) + t^2\mathcal{A}(UV^\mathsf{T}), \mathcal{A}(UR^\mathsf{T} + LV^\mathsf{T})\rangle \\
&\qquad + \langle \Delta, 2t\mathcal{A}(UV^\mathsf{T})\rangle + \langle t\mathcal{A}(UR^\mathsf{T} + LV^\mathsf{T}) + t^2\mathcal{A}(UV^\mathsf{T}), 2t\mathcal{A}(UV^\mathsf{T})\rangle \\
&\qquad - \langle \Delta, \mathcal{A}(UR^\mathsf{T} + LV^\mathsf{T})\Big) \\
&= \lim_{t\to 0} \frac{2}{t}\Big(\langle t\mathcal{A}(UR^\mathsf{T} + LV^\mathsf{T}) + t^2\mathcal{A}(UV^\mathsf{T}), \mathcal{A}(UR^\mathsf{T} + LV^\mathsf{T})\rangle \\
&\qquad + \langle \Delta, 2t\mathcal{A}(UV^\mathsf{T})\rangle + \langle t\mathcal{A}(UR^\mathsf{T} + LV^\mathsf{T}) + t^2\mathcal{A}(UV^\mathsf{T}), 2t\mathcal{A}(UV^\mathsf{T})\rangle\Big) \\
&= \lim_{t\to 0} \frac{2}{t}\Big(t\langle \mathcal{A}(UR^\mathsf{T} + LV^\mathsf{T}) + t\mathcal{A}(UV^\mathsf{T}), \mathcal{A}(UR^\mathsf{T} + LV^\mathsf{T})\rangle \\
&\qquad + 2t\langle \Delta, \mathcal{A}(UV^\mathsf{T})\rangle + 2t^2\langle \mathcal{A}(UR^\mathsf{T} + LV^\mathsf{T}) + t^2\mathcal{A}(UV^\mathsf{T}), \mathcal{A}(UV^\mathsf{T})\rangle\Big).
\end{aligned}$$

Taking the limit as $t \to 0$,

$$\begin{aligned}
\nabla^2 f(L,R)[U,V] &= 2\Big(\langle \mathcal{A}(UR^\mathsf{T} + LV^\mathsf{T}), \mathcal{A}(UR^\mathsf{T} + L^\mathsf{T})\rangle + 2\langle \Delta, \mathcal{A}(UV^\mathsf{T})\rangle\Big) \\
&= 2\Big(\|\mathcal{A}(UR^\mathsf{T} + LV^\mathsf{T})\|_2^2 + 2\langle \mathcal{A}(LR^\mathsf{T}) - b, \mathcal{A}(UV^\mathsf{T})\rangle\Big) \\
&= 2\|\mathcal{A}(UR^\mathsf{T} + LV^\mathsf{T})\|_2^2 + 4\langle \mathcal{A}(LR^\mathsf{T}) - b, \mathcal{A}(UV^\mathsf{T})\rangle
\end{aligned}$$

which yields the result. ∎

We can prove Lemma E.3, which bounds the Lipschitz constant of the Hessian of $f(\cdot)$ as defined in Eq. E.6.

Proof of Lemma E.3. Given some vector $v \in \mathbb{R}^{(d_1+d_2)k}$, group the entries of the vector v into two matrices: let $v = (U, V)$ for matrices $U \in \mathbb{R}^{d_1 \times k}, V \in \mathbb{R}^{d_2 \times k}$. Then the quadratic form of the Hessian is

$$\langle v, \nabla^2 f(\theta) v \rangle = D^2 f(L, R)[U, V].$$

The objective $f(\cdot)$ satisfies Assumption 2 with respect to Lipschitz constant H if $\forall \theta_1, \theta_2, v \in \mathbb{R}^{(d_1+d_2)k}$,

$$|\langle v, (\nabla^2 f(\theta_1) - \nabla^2 f(\theta_2)) v \rangle| \le H \|v\|_2^2 \|\theta_1 - \theta_2\|_2.$$

For any pair $\theta_1, \theta_2 \in \mathbb{R}^{(d_1+d_2)k}$, denote the entries by $\theta_1 = (L_1, R_1)$ and $\theta_2 = (L_2, R_2)$, and for any $v \in \mathbb{R}^{(d_1+d_2)k}$, denote the entries by $v = (U, V)$ as above. Then the above inequality is equivalent to

$$|D^2 f(L_1, R_1)[U, V] - D^2 f(L_2, R_2)[U, V]| \le H \left(\|U\|_{\mathsf{F}}^2 + \|V\|_{\mathsf{F}}^2 \right) \|(L_1, R_1) - (L_2, R_2)\|_2.$$

where

$$\|(L, R)\|_2 \stackrel{\text{def}}{=} \sqrt{\|L\|_{\mathsf{F}}^2 + \|R\|_{\mathsf{F}}^2} = \|\theta\|_2.$$

By Lemma E.4, for any $\theta = (L, R)$ and any $v = (U, V)$, the Hessian satisfies

$$D^2 f(L, R)[U, V] = 2\|\mathcal{A}(UR^{\mathsf{T}} + LV^{\mathsf{T}})\|_2^2 + 4\langle \mathcal{A}(LR^{\mathsf{T}}) - b, \mathcal{A}(UV^{\mathsf{T}}) \rangle.$$

By linearity of $\mathcal{A}(\cdot)$, we can expand the squared norm:

$$\begin{aligned} D^2 f(L, R)[U, V] &= 2\|\mathcal{A}(UR^{\mathsf{T}}) + \mathcal{A}(LV^{\mathsf{T}})\|_2^2 + 4\langle \mathcal{A}(LR^{\mathsf{T}}) - b, \mathcal{A}(UV^{\mathsf{T}}) \rangle \\ &= 2\Big(\|\mathcal{A}(LV^{\mathsf{T}})\|_2^2 + \|\mathcal{A}(UR^{\mathsf{T}})\|_2^2 \Big) + 4\langle \mathcal{A}(UR^{\mathsf{T}}), \mathcal{A}(LV^{\mathsf{T}}) \rangle + 4\langle \mathcal{A}(LR^{\mathsf{T}}) - b, \mathcal{A}(UV^{\mathsf{T}}) \rangle. \end{aligned}$$

Thus

$$\begin{aligned} D^2 f(L_1, R_1)[U, V] - D^2 f(L_2, R_2)[U, V] &= 2 \left(\|\mathcal{A}(L_1 V^{\mathsf{T}})\|_2^2 - \|\mathcal{A}(L_2 V^{\mathsf{T}})\|_2^2 \right) && \text{(E.7)} \\ &\quad + 2 \left(\|\mathcal{A}(UR_1^{\mathsf{T}})\|_2^2 - \|\mathcal{A}(UR_2^{\mathsf{T}})\|_2^2 \right) && \text{(E.8)} \\ &\quad + 4 \left(\langle \mathcal{A}(UR_1^{\mathsf{T}}), \mathcal{A}(L_1 V^{\mathsf{T}}) \rangle - \langle \mathcal{A}(UR_2^{\mathsf{T}}), \mathcal{A}(L_2 V^{\mathsf{T}}) \rangle \right) && \text{(E.9)} \\ &\quad + 4\langle \mathcal{A}(L_1 R_1^{\mathsf{T}}) - \mathcal{A}(L_2 R_2^{\mathsf{T}}), \mathcal{A}(UV^{\mathsf{T}}) \rangle. && \text{(E.10)} \end{aligned}$$

We bound the magnitude of each term in sequence. For the first term in Line E.7, we observe

$$\begin{aligned} \left| \|\mathcal{A}(L_1 V^{\mathsf{T}})\|_2^2 - \|\mathcal{A}(L_2 V^{\mathsf{T}})\|_2^2 \right| &= |\langle \mathcal{A}(L_1 V^{\mathsf{T}}) + \mathcal{A}(L_2 V^{\mathsf{T}}), \mathcal{A}(L_1 V^{\mathsf{T}}) - \mathcal{A}(L_2 V^{\mathsf{T}}) \rangle| \\ &= |\langle \mathcal{A}\left((L_1 + L_2) V^{\mathsf{T}} \right), \mathcal{A}\left((L_1 - L_2) V^{\mathsf{T}} \right) \rangle| \\ &\le \|\mathcal{A}\left((L_1 + L_2) V^{\mathsf{T}} \right)\|_2 \|\mathcal{A}\left((L_1 - L_2) V^{\mathsf{T}} \right)\|_2. \end{aligned}$$

The operator norm of $\mathcal{A}(\cdot)$ can be bounded as

$$\|\mathcal{A}(X)\|_2^2 = \sum_{i=1}^m \langle A_i, X \rangle^2 \le \|X\|_{\mathsf{F}}^2 \sum_{i=1}^m \|A_i\|_{\mathsf{F}}^2.$$

Thus

$$\|\mathcal{A}\|_2 \leq \left(\sum_{i=1}^{m} \|A_i\|_{\mathsf{F}}^2\right)^{1/2}. \tag{E.11}$$

We can thus bound the term in Line E.7 as

$$\begin{aligned}\left|\|\mathcal{A}(L_1V^\mathsf{T})\|_2^2 - \|\mathcal{A}(L_2V^\mathsf{T})\|_2^2\right| &\leq \|\mathcal{A}\left((L_1+L_2)V^\mathsf{T}\right)\|_2 \|\mathcal{A}\left((L_1-L_2)V^\mathsf{T}\right)\|_2 \\ &\leq \|\mathcal{A}\|_2^2 \|(L_1+L_2)V^\mathsf{T}\|_{\mathsf{F}} \|(L_1-L_2)V^\mathsf{T}\|_{\mathsf{F}} \\ &\leq \|\mathcal{A}\|_2^2 \|L_1+L_2\|_{\mathsf{F}} \|V\|_{\mathsf{F}} \|L_1-L_2\|_{\mathsf{F}} \|V\|_{\mathsf{F}}.\end{aligned}$$

In the ball $\|\theta\|_2 \leq B$, we have that for $\theta = (L, R)$

$$\|L\|_{\mathsf{F}} \leq \|\theta\|_2 \leq B.$$

Thus $\|L_1 + L_2\|_{\mathsf{F}} \leq 2B$, so we can bound

$$\begin{aligned}\left|\|\mathcal{A}(L_1V^\mathsf{T})\|_2^2 - \|\mathcal{A}(L_2V^\mathsf{T})\|_2^2\right| &\leq 2B\|\mathcal{A}\|_2^2 \|V\|_F^2 \|L_1 - L_2\|_{\mathsf{F}} \\ &\leq 2B\|\mathcal{A}\|_2^2 \|V\|_{\mathsf{F}}^2 \|(L_1,R_1) - (L_2,R_2)\|_2\end{aligned}$$

where the last line follows by

$$\|(L_1,R_1) - (L_2,R_2)\|_2 \stackrel{\text{def}}{=} \sqrt{\|L_1-L_2\|_{\mathsf{F}}^2 + \|R_1-R_2\|_{\mathsf{F}}^2} \geq \|L_1 - L_2\|_{\mathsf{F}}.$$

An analogous argument bounds the term in Line E.8 as

$$\left|\|\mathcal{A}(UR_1^\mathsf{T})\|_2^2 - \|\mathcal{A}(UR_2^\mathsf{T})\|_2^2\right| \leq 2B\|\mathcal{A}\|_2^2 \|U\|_{\mathsf{F}}^2 \|(L_1,R_1) - (L_2,R_2)\|_2.$$

To bound the term in Line E.9, we first observe that

$$\begin{aligned}&\langle \mathcal{A}(UR_1^\mathsf{T}), \mathcal{A}(L_1V^\mathsf{T})\rangle - \langle \mathcal{A}(UR_2^\mathsf{T}), \mathcal{A}(L_2V^\mathsf{T})\rangle \\ &= \langle \mathcal{A}(UR_1^\mathsf{T}), \mathcal{A}(L_1V^\mathsf{T})\rangle - \langle \mathcal{A}(L_1V^\mathsf{T}), \mathcal{A}(UR_2^\mathsf{T})\rangle \\ &\quad + \langle \mathcal{A}(L_1V^\mathsf{T}), \mathcal{A}(UR_2^\mathsf{T})\rangle - \langle \mathcal{A}(UR_2^\mathsf{T}), \mathcal{A}(L_2V^\mathsf{T})\rangle \\ &= \langle \mathcal{A}(UR_1^\mathsf{T}) - \mathcal{A}(UR_2^\mathsf{T}), \mathcal{A}(L_1V^\mathsf{T})\rangle + \langle \mathcal{A}(L_1V^\mathsf{T}) - \mathcal{A}(L_2V^\mathsf{T}), \mathcal{A}(UR_2^\mathsf{T})\rangle \\ &= \langle \mathcal{A}\left(U(R_1-R_2)^\mathsf{T}\right), \mathcal{A}(L_1V^\mathsf{T})\rangle + \langle \mathcal{A}\left((L_1-L_2)V^\mathsf{T}\right), \mathcal{A}(UR_2^\mathsf{T})\rangle\end{aligned}$$

Thus

$$\begin{aligned}&|\langle \mathcal{A}(UR_1^\mathsf{T}), \mathcal{A}(L_1V^\mathsf{T})\rangle - \langle \mathcal{A}(UR_2^\mathsf{T}), \mathcal{A}(L_2V^\mathsf{T})\rangle| \\ &= |\langle \mathcal{A}\left(U(R_1-R_2)^\mathsf{T}\right), \mathcal{A}(L_1V^\mathsf{T})\rangle + \langle \mathcal{A}\left((L_1-L_2)V^\mathsf{T}\right), \mathcal{A}(UR_2^\mathsf{T})\rangle| \\ &\leq \|\mathcal{A}\left(U(R_1-R_2)^\mathsf{T}\right)\|_2 \|\mathcal{A}(L_1V^\mathsf{T})\|_2 + \|\mathcal{A}\left((L_1-L_2)V^\mathsf{T}\right)\|_2 \|\mathcal{A}(UR_2^\mathsf{T})\|_2 \\ &\leq \|\mathcal{A}\|_2^2 \|U(R_1-R_2)^\mathsf{T}\|_{\mathsf{F}} \|L_1V^\mathsf{T}\|_{\mathsf{F}} + \|\mathcal{A}\|_2^2 \|(L_1-L_2)V^\mathsf{T}\|_{\mathsf{F}} \|UR_2^\mathsf{T}\|_{\mathsf{F}} \\ &\leq \|\mathcal{A}\|_2^2 \|U\|_{\mathsf{F}} \|R_1-R_2\|_{\mathsf{F}} \|L_1\|_{\mathsf{F}} \|V\|_{\mathsf{F}} + \|\mathcal{A}\|_2^2 \|L_1-L_2\|_{\mathsf{F}} \|V\|_{\mathsf{F}} \|U\|_{\mathsf{F}} \|R_2\|_{\mathsf{F}} \\ &= \|\mathcal{A}\|_2^2 \|U\|_{\mathsf{F}} \|V\|_{\mathsf{F}} \left(\|R_1-R_2\|_{\mathsf{F}} \|L_1\|_{\mathsf{F}} + \|L_1-L_2\|_{\mathsf{F}} \|R_2\|_{\mathsf{F}}\right)\end{aligned}$$

In the ball $\|\theta\|_2 \leq B$, we have that for $\|L_1\|_\mathsf{F} \leq B$ and $\|R_2\|_\mathsf{F} \leq B$. Thus

$$\begin{aligned}|\langle \mathcal{A}(UR_1^\mathsf{T}), \mathcal{A}(L_1V^\mathsf{T})\rangle - \langle \mathcal{A}(UR_2^\mathsf{T}), \mathcal{A}(L_2V^\mathsf{T})\rangle| \\ \leq B\|\mathcal{A}\|_2^2\|U\|_\mathsf{F}\|V\|_\mathsf{F}\left(\|R_1 - R_2\|_\mathsf{F} + \|L_1 - L_2\|_\mathsf{F}\right).\end{aligned}$$

Recall that for any $a, b \in R$, it holds that $a + b \leq 2\sqrt{a^2 + b^2}$ and $2ab \leq a^2 + b^2$. Thus

$$\|R_1 - R_2\|_\mathsf{F} + \|L_1 - L_2\|_\mathsf{F} \leq 2\sqrt{\|L_1 - L_2\|_\mathsf{F}^2 + \|R_1 - R_2\|_\mathsf{F}^2} = 2\|(L_1, R_1) - (L_2, R_2)\|_2$$

and

$$\|U\|_\mathsf{F}\|V\|_\mathsf{F} \leq \frac{1}{2}\left(\|U\|_\mathsf{F}^2 + \|V\|_\mathsf{F}^2\right).$$

Combining these bounds yields

$$\begin{aligned}|\langle \mathcal{A}(UR_1^\mathsf{T}), \mathcal{A}(L_1V^\mathsf{T})\rangle - \langle \mathcal{A}(UR_2^\mathsf{T}), \mathcal{A}(L_2V^\mathsf{T})\rangle| \\ \leq B\|\mathcal{A}\|_2^2\left(\|U\|_\mathsf{F}^2 + \|V\|_\mathsf{F}^2\right)\|(L_1, R_1) - (L_2, R_2)\|_2.\end{aligned}$$

Lastly, we bound the term in Line E.10. We begin by observing that

$$\begin{aligned}\mathcal{A}(L_1R_1^\mathsf{T}) - \mathcal{A}(L_2R_2^\mathsf{T}) &= \mathcal{A}(L_1R_1^\mathsf{T}) - \mathcal{A}(L_1R_2^\mathsf{T}) + \mathcal{A}(L_1R_2^\mathsf{T}) - \mathcal{A}(L_2R_2^\mathsf{T}) \\ &= \mathcal{A}\left(L_1(R_1 - R_2)^\mathsf{T}\right) + \mathcal{A}\left((L_1 - L_2)R_2^\mathsf{T}\right)\end{aligned}$$

Thus

$$\begin{aligned}|\langle \mathcal{A}(L_1R_1^\mathsf{T}) - \mathcal{A}(L_2R_2^\mathsf{T}), \mathcal{A}(UV^\mathsf{T})\rangle| &= |\langle \mathcal{A}\left(L_1(R_1 - R_2)^\mathsf{T}\right) + \mathcal{A}\left((L_1 - L_2)R_2^\mathsf{T}\right), \mathcal{A}(UV^\mathsf{T})\rangle| \\ &= |\langle \mathcal{A}\left(L_1(R_1 - R_2)^\mathsf{T}\right), \mathcal{A}(UV^\mathsf{T})\rangle + \langle \mathcal{A}\left((L_1 - L_2)R_2^\mathsf{T}\right), \mathcal{A}(UV^\mathsf{T})\rangle| \\ &\leq \|\mathcal{A}\left(L_1(R_1 - R_2)^\mathsf{T}\right)\|_2\|\mathcal{A}(UV^\mathsf{T})\|_2 + \|\mathcal{A}\left((L_1 - L_2)R_2^\mathsf{T}\right)\|_2\|\mathcal{A}(UV^\mathsf{T})\|_2 \\ &\leq \|\mathcal{A}\|_2^2\|L_1(R_1 - R_2)^\mathsf{T}\|_F\|UV^\mathsf{T}\|_F + \|\mathcal{A}\|_2^2\|(L_1 - L_2)R_2^\mathsf{T}\|_F\|UV^\mathsf{T}\|_F \\ &\leq \|\mathcal{A}\|_2^2\left(\|L_1\|_\mathsf{F}\|R_1 - R_2\|_\mathsf{F}\|U\|_\mathsf{F}\|V\|_\mathsf{F} + \|L_1 - L_2\|_\mathsf{F}\|R_2\|_\mathsf{F}\|U\|_\mathsf{F}\|V\|_\mathsf{F}\right) \\ &= \|\mathcal{A}\|_2^2\|U\|_\mathsf{F}\|V\|_\mathsf{F}\left(\|L_1\|_\mathsf{F}\|R_1 - R_2\|_\mathsf{F} + \|L_1 - L_2\|_\mathsf{F}\|R_2\|_\mathsf{F}\right).\end{aligned}$$

In the ball $\|\theta\|_2 \leq B$, we have that for $\|L_1\|_\mathsf{F} \leq B$ and $\|R_2\|_\mathsf{F} \leq B$. Thus

$$|\langle \mathcal{A}(L_1R_1^\mathsf{T}) - \mathcal{A}(L_2R_2^\mathsf{T}), \mathcal{A}(UV^\mathsf{T})\rangle| \leq B\|\mathcal{A}\|_2^2\|U\|_\mathsf{F}\|V\|_\mathsf{F}\left(\|R_1 - R_2\|_\mathsf{F} + \|L_1 - L_2\|_\mathsf{F}\right).$$

Recalling the bounds $\|V\|_\mathsf{F}\|U\|_\mathsf{F}$ and $(\|L_1 - L_2\|_\mathsf{F} + \|R_1 - R_2\|_\mathsf{F})$ established above, we have

$$|\langle \mathcal{A}(L_1R_1^\mathsf{T}) - \mathcal{A}(L_2R_2^\mathsf{T}), \mathcal{A}(UV^\mathsf{T})\rangle| \leq B\|\mathcal{A}\|_2^2\left(\|U\|_\mathsf{F}^2 + \|V\|_\mathsf{F}^2\right)\|(L_1, R_1) - (L_2, R_2)\|_2.$$

Employing triangle inequality and each of the above bounds implies

$$\begin{aligned}\left|D^2 f(L_1, R_1)[U, V] - D^2 f(L_2, R_2)[U, V]\right| &\leq 2 \cdot 2B\|\mathcal{A}\|_2^2\|V\|_\mathsf{F}^2\|(L_1, R_1) - (L_2, R_2)\|_2 \\ &\quad + 2 \cdot 2B\|\mathcal{A}\|_2^2\|U\|_\mathsf{F}^2\|(L_1, R_1) - (L_2, R_2)\|_2 \\ &\quad + 4B\|\mathcal{A}\|_2^2\left(\|U\|_\mathsf{F}^2 + \|V\|_\mathsf{F}^2\right)\|(L_1, R_1) - (L_2, R_2)\|_2 \\ &\quad + 4B\|\mathcal{A}\|_2^2\left(\|U\|_\mathsf{F}^2 + \|V\|_\mathsf{F}^2\right)\|(L_1, R_1) - (L_2, R_2)\|_2 \\ &\leq 12B\|\mathcal{A}\|_2^2\left(\|U\|_\mathsf{F}^2 + \|V\|_\mathsf{F}^2\right)\|(L_1, R_1) - (L_2, R_2)\|_2\end{aligned}$$

which implies the result. ∎

E.2. Deferred Proofs from Section A.2.

Proof of Lemma A.11. Observe that for any $\theta \in \mathbb{R}^d$,

$$\nabla f(\theta) = A^\mathsf{T}(A\theta - y) = A^\mathsf{T}(A\theta - A\theta^* - A\eta) = A^\mathsf{T}A(\theta - \theta^*) + A\eta.$$

Thus for ρ a standard Gaussian,

$$\begin{aligned}\mathrm{EGOP}(f) &= \mathbb{E}_{\theta\sim\rho}\left[\left(A^\mathsf{T}(A\theta - y)\right)\left(A^\mathsf{T}(A\theta - y)\right)^\mathsf{T}\right] \\ &= A^\mathsf{T}\mathbb{E}_{\theta\sim\rho}\left[A\theta\theta^\mathsf{T}A + yy^\mathsf{T}\right]A \\ &= \left(A^\mathsf{T}A\right)^2 + (A^\mathsf{T}y)(A^\mathsf{T}y)^\mathsf{T}\end{aligned}$$

using the fact that ρ is mean-zero and isotropic. Thus $\mathrm{EGOP}(f)$ is a rank-1 perturbation of the matrix $(A^\mathsf{T}A)^2$, which is itself PSD and has eigenvalues $\sigma_k(A)^4$. Moreover if we let $A = Q_1\Sigma Q_2^\mathsf{T}$ denote the SVD of A, we can rewrite

$$\mathrm{EGOP}(f) = Q_2\left(\Sigma^4 + \Sigma Q_1^\mathsf{T}yy^\mathsf{T}Q_1\Sigma\right)V^\mathsf{T}$$

where Σ^4 is diagonal and has entries $\{\sigma_i^4(A)\}_{i=1}^n$. By Golub (1973) [16], the eigenvalues of $\mathrm{EGOP}(f)$ thus satisfy

$$\sigma_i^4(A) \geq \lambda_i(\mathrm{EGOP}(f)) \geq \sigma_{i+1}^4(A)\ \forall i \in [2,\ldots,n]$$

and

$$\sigma_1^4(A) + \|\Sigma Q_1^\mathsf{T}y\|_2^2 \geq \lambda_1(\mathrm{EGOP}) \geq \sigma_2^4(A)$$

which implies the result. ∎

Appendix F. Extended Discussion of Related Works.

In this section, we expand on the overview of related work presented in Section 1.1.

Geometric sensitivity of adaptive methods. A large body of research has been devoted to understanding the settings in which the per-coordinate learning rates of adaptive algorithms confer a benefit over more basic methods, such as (S)GD. Recently there has been renewed interest in distinguishing the properties of adaptive algorithms versus SGD because several empirical studies suggest that adaptive methods outperform SGD when training transformer models [23, 42]. Traditional analyses of Adagrad in the context of online convex optimization establish regret bounds which can be either better or worse than those enjoyed by SGD by up to a factor of $\sqrt{d}$, for d the number of problem parameters [4, 14]. More recent research has studied the rates at which adaptive methods converge to stationary points. Several works study convergence guarantees, measured in terms of the ℓ_2 norm of the gradient, on smooth non-convex objectives [11, 38]. These results show that Adagrad, Adam and related variants achieve rates matching those enjoyed by SGD, while having the key distinction that Adagrad and its variants do not require a-priori knowledge of the objective function's Lipschitz constant [11, 38].

In order to shed light on the question of when adaptive algorithms enjoy provably stronger guarantees than SGD, a recent line of work studies convergence under more refined geometric assumptions, with particular emphasis on assumptions that are *not* rotationally invariant [21, 25, 40]. Xie et al. [40] establish convergence guarantees in terms of the L_∞ smoothness constant of the objective and also in terms of the related Hessian 1-norm. They show that rotationally invariant geometric assumptions do not suffice to capture settings when Adam out-performs SGD through experiments examining the sensitivity of Adam to orthonormal rotations [40].

Jiang et al. [21] and Liu et al. [25] study convergence of Adagrad on objectives that satisfy coordinate-wise smoothness. Both works prove similar convergence guarantees for Adagrad in terms of ℓ_1 gradient norm, rather than the ℓ_2 gradient norm measure more widely studied; Jiang et al. [21] establish convergence guarantees showing that SGD remains worst-case optimal even in the setting of coordinate-wise smoothness when measuring stationarity with the ℓ_2 gradient norm, motivating the need to measure stationarity with the ℓ_1 norm in order to prove separation between Adagrad and SGD. Using this ℓ_1 stationarity measure, Jiang et al. [21] establish provable separation between SGD's and Adagrad's convergence to stationary points of non-convex objectives. They show that when objective functions exhibit certain geometric properties, measured by their coordinate-wise smoothness constants, Adagrad's upper bounds are lower than SGD's lower bounds by a factor of d [21]. Our analysis builds on that of Jiang et al. [21] and Liu et al. [25], as a consequence of our results is that the EGOP reparameterization proposed in this work acts to transform objectives into the setting identified by Jiang et al. [21] where Adagrad's convergence guarantees compare most favorably with SGD's.

Ling et al. [24] develop a rotationally equivariant extension of Adam, termed VectorAdam, in an effort to reduce axis-aligned artifacts that arise when using adaptive methods to solve geometric optimization problems such as differentiable rendering. This method targets applications when the problem parameters $\theta \in \mathbb{R}^{r \cdot n}$ represent a collection of r different vectors in $\mathbb{R}^n$. VectorAdam uses the squared gradient norm of each n-dimensional vector to rescale the learning rates of all entries in each of the r vectors comprising θ, making the algorithm equivariant to transformations of the form $Q \operatorname{mat} \theta$, where $\operatorname{mat} \theta \in \mathbb{R}^{n \times r}$ is the reshaping of $\theta \in \mathbb{R}^{r \cdot n}$ and $Q \in \mathbb{R}^{n \times n}$ is orthonormal [24].

Change-of-basis for Optimization. Several recent works propose that when using Adam and its variants to train neural networks, different choices of orthonormal transformation can reduce the computational cost associated with these algorithms and improve their performance [27, 37, 43]. Gupta et al. [17] introduced Shampoo, an efficient preconditioning method for optimizing tensor spaces. Vyas et al. [37] formalized a connection between Shampoo and a variant of Adagrad, leading to a new proposed method called SOAP. Designed for training neural networks, SOAP computes an orthonormal reparameterization based on the singular vectors of the matrix-valued network gradients and performs optimization in this basis. Vyas et al. [37] empirically examine the performance of SOAP and find that it outperforms both Adam and Shampoo in LLM pre-training. Zhao et al. [43] propose GaLore, a method that simultaneously performs reparameterization and dimensionality reduction. GaLore computes a similar orthogonal basis to that used in SOAP, but instead of a full-dimensional change-of-basis GaLore retains only leading basis vectors in order to reduce dimension [43]. Maes

et al. [27] empirically study Adam's rotational sensitivity and examine the power of existing geometric assumptions (such as those leveraged by Xie et al. [40]) to explain Adam's performance when training transformer architectures. They propose an orthonormal reparameterization, similar to those used by SOAP and GaLore, and show empirically that this can improve Adam's performance [27].

SOAP and GaLore both call for periodically re-computing the change-of-basis matrices. In their experiments, the fewest re-computations performed by Vyas et al. [37] was once every 100 batches, and they show that the performance gap between SOAP and Adam narrows as the number of batches between re-computations increases. In contrast, our we show that with our proposed choice of basis, a single up-front computation of the change-of-basis suffices to improve the performance of both Adam and Adagrad in a variety of settings, as shown in Section 6.

Outside of the domain neural network training, several works have considered data-driven methods for performing dimensionality reduction when optimizing objectives with low-rank EGOP matrices, including the works of Cartis et al. [3] and Cosson et al. [7]. For functions with low-dimensional active subspaces, Cartis et al. [3] study using the EGOP eigenbasis to reparameterize and reduce the dimension of optimization problems. They demonstrate that this approach yields computational speedups for loss functions whose EGOP matrix is low-rank. Cosson et al. [7] develop gradient descent algorithms which leverage the EGOP eigenbasis. Their method first estimates the EGOP and computes the corresponding leading r-eigenvectors, and then performs gradient descent steps along the directions of these r vectors. In the setting of exactly low-rank functions, they prove convergence results illustrating that this approach improves the dimensional dependency of gradient descent.

Appendix G. Kronecker Product Constraints in SOAP/Shampoo.

In this section, we provide a detailed comparison of the change of basis employed by EGOP reparameterization and those performed by SOAP and Shmpoo [37, 17]. We instantiate an example of EGOP reparameterization for an objective whose parameters can be viewed as matrices, as this highlights some of the conceptual differences between the methods.

In many key applications of adaptive algorithms, including training neural networks, the parameters over which optimization occurs are matrix-valued. As a simple illustration, consider optimizing a single-layer linear fully-connected network. Let n_{in} denote the number of input features to the layer, and n_{out} denote the number of output features. This network then has $n_{\text{in}} \cdot n_{\text{out}}$ parameters, which can be expressed either as a vector, $\theta \in \mathbb{R}^{n_{\text{in}} n_{\text{out}}}$, or as a matrix, denoted $\text{mat}(\theta) \in \mathbb{R}^{n_{\text{in}} \times n_{\text{out}}}$.

Denote the training data and labels by $A \in \mathbb{R}^{n_{\text{in}} \times n_{\text{samples}}}$ and $Y \in \mathbb{R}^{n_{\text{out}} \times n_{\text{samples}}}$, and consider minimizing loss function

$$f(\theta) = \frac{1}{2} \|\text{mat}(\theta)^{\mathsf{T}} A - Y\|_{\mathsf{F}}^2.$$

Similarly, the vector-valued gradients $\nabla f(\theta) \in \mathbb{R}^{n_{\text{in}} n_{\text{out}}}$ can also be viewed as matrices, $\text{mat}(\nabla f(\theta)) \in \mathbb{R}^{n_{\text{in}} \times n_{\text{out}}}$.

In the method proposed in this work, we consider gradients to be vector-valued when forming the EGOP. Thus for this single-layer objective, $\text{EGOP}(f) \in \mathbb{R}^{n_{\text{in}} n_{\text{out}} \times n_{\text{in}} n_{\text{out}}}$. We emphasize this vector-view of gradients for clarity, because the EGOP matrix has distinct eigenvectors

from the expectation of the matrix product, $\text{mat}(\nabla f(\theta))\,\text{mat}(\nabla f(\theta))^\mathsf{T}$. The related methods discussed in the introduction, including SOAP and GaLore, consider transformations by the matrices

$$Q_L = \text{eigenvectors}(\mathbb{E}[\text{mat}\,(\nabla f(\theta))\,\text{mat}\,(\nabla f(\theta))^\mathsf{T}])$$
$$Q_R = \text{eigenvectors}(\mathbb{E}[\text{mat}\,(\nabla f(\theta))^\mathsf{T}\,\text{mat}\,(\nabla f(\theta))])$$

and closely related transformations [27, 37, 43]. In general the eigenvectors in Q_L, Q_R correspond to different transformations than the eigenvectors of the EGOP. In particular, letting V denote the eigenbasis of the EGOP formed from vector-valued gradients, $V \neq Q_L \otimes Q_R$. Moreover, the class of orthonormal matrices obtainable from the EGOP eigenbasis is strictly more general than the class $Q_L \otimes Q_R$ for pairs of orthogonal matrices Q_L, Q_R, as formalized below:

Lemma G.1. *For any orthogonal matrices $Q_L \in \mathbb{R}^{n_{in}\times n_{in}}, Q_R \in \mathbb{R}^{n_{out}\times n_{out}}$ and any objective function $f : \mathbb{R}^{n_{in}n_{out}} \to \mathbb{R}$, there exists orthogonal $V \in \mathbb{R}^{n_{in}n_{out}\times n_{in}n_{out}}$ such that*

$$\forall\theta \in \mathbb{R}^{n_{in}n_{out}}, \quad Q_L^\mathsf{T}\,\text{mat}(\nabla f(\theta))Q_R = \text{mat}(V\nabla f(\theta)).$$

However, there exist values of n_{in}, n_{out}, objective functions $f : \mathbb{R}^{n_{in}n_{out}} \to \mathbb{R}$, and orthogonal matrices $V \in \mathbb{R}^{n_{in}n_{out}\times n_{in}n_{out}}$ such that no orthogonal matrices $Q_L \in \mathbb{R}^{n_{in}\times n_{in}}, Q_R \in \mathbb{R}^{n_{out}\times n_{out}}$ satisfy

$$\forall\theta \in \mathbb{R}^{n_{in}n_{out}}, \quad Q_L^\mathsf{T}\,\text{mat}(\nabla f(\theta))Q_R = \text{mat}(V\nabla f(\theta)).$$

Proof. First we show that for any Q_L, Q_R, there exists suitable V satisfying the property. We begin by noting that for any matrices A, B, C of compatible dimension,

$$\text{vec}(ABC) = (C^\mathsf{T} \otimes A)\,\text{vec}(B).$$

Thus for any Q_L, Q_R,

$$\text{vec}(Q_L^\mathsf{T}\,\text{mat}(\nabla f(\theta))Q_R) = (Q_R^\mathsf{T} \otimes Q_L^\mathsf{T})\nabla f(\theta)$$

where the second equality uses the fact that trivially $\text{vec}(\text{mat}(\nabla f(\theta))) = \nabla f(\theta)$. The Kronecker product of orthogonal matrices is orthogonal, so choice of $V = (Q_R^\mathsf{T} \otimes Q_L^\mathsf{T})$ satisfies the desired property.

We now show there exist orthogonal matrices $V \in \mathbb{R}^{n_{\text{in}}n_{\text{out}}\times n_{\text{in}}n_{\text{out}}}$ such that no orthogonal matrices $Q_L \in \mathbb{R}^{n_{\text{in}}\times n_{\text{in}}}, Q_R \in \mathbb{R}^{n_{\text{out}}\times n_{\text{out}}}$ satisfy

$$Q_R \otimes Q_L^\mathsf{T} = V.$$

This is a consequence of the fact that not all orthogonal matrices admit Kronecker product factorizations. Here we give one specific construction.

Let $\vec{1}_d$ denote the all-ones vector in $\mathbb{R}^d$. Consider V with leading column

$$v_1 = \vec{1}_{n_{\text{in}}n_{\text{out}}}/\sqrt{n_{\text{in}}n_{\text{out}}}$$

and second column

$$v_2 = (n_{\text{in}}n_{\text{out}})^{-1/2} \cdot \text{concatenate}(\vec{1}_{n_{\text{in}}n_{\text{out}}/2}, -\vec{1}_{n_{\text{in}}n_{\text{out}}/2})$$

. Given the entries of v_1 are identical, this implies all the entries in the first column of Q_R are identical, and thus that the first n_{in} columns of V comprise concatenated copies of Q_L^T. However this contradicts the fact that the entries of v_2 are identical in magnitude and nonzero but have positive sign for the first $n_{\text{in}}n_{\text{out}}/2$ entries and negative sign for the rest. Thus no orthogonal matrices V with such first and second columns can be decomposed into $Q_R \otimes Q_L^T$, so in particular there exists some vector $z \in \mathbb{R}^{n_{\text{in}}n_{\text{out}}}$ such that

$$Q_R \otimes Q_L^\intercal z \neq Vz.$$

Thus for any $f(\cdot)$ such that there exists θ with $\nabla f(\theta) = z$, it holds that for this value

$$Q_L^\intercal \,\text{mat}(\nabla f(\theta))Q_R = \text{mat}\left((Q_R \otimes Q_L^\intercal)\nabla f(\theta)\right) \neq \text{mat}(V\nabla f(\theta)).$$

■