LoRA meets Riemannion: Muon Optimizer for Parametrization-independent Low-Rank Adapters

Anonymous authors
Paper under double-blind review

Abstract

This work presents a novel, fully Riemannian framework for Low-Rank Adaptation (LoRA) that geometrically treats low-rank adapters by optimizing them directly on the fixed-rank manifold. This formulation eliminates the parametrization ambiguity present in standard Euclidean optimizers. Our framework integrates three key components to achieve this: (1) we derive Riemannion, a new Riemannian optimizer on the fixed-rank matrix manifold that generalizes the recently proposed Muon optimizer; (2) we develop a Riemannian gradient-informed LoRA initialization, and (3) we provide an efficient implementation without prominent overhead that uses automatic differentiation to compute arising geometric operations while adhering to best practices in numerical linear algebra. Comprehensive experimental results on both LLM and diffusion model architectures demonstrate that our approach yields consistent and noticeable improvements in convergence speed and final task performance over both standard LoRA and its state-of-the-art modifications.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities across a wide range of natural language processing tasks Brown et al. (2020); Touvron et al. (2023a;b). However, the computational and storage costs associated with training and deploying such models at scale pose significant challenges. To reduce these costs, parameter-efficient fine-tuning techniques such as low-rank adaptation (LoRA) Hu et al. (2022) have emerged as a practical solution. LoRA enables efficient adaptation of pre-trained models by embedding learnable low-rank matrices into specific weight updates, allowing most of the original parameters to remain frozen. In particular, the main idea of LoRA is to fine-tune a pretrained model using a rank-r correction matrix ΔW:

$$W + \Delta W = W + AB^\top, \quad A \in \mathbb{R}^{m\times r}, \quad B \in \mathbb{R}^{n\times r},$$

where W remains constant during training and A, B are optimized via gradient-based optimization methods.

Despite its efficiency, the dominant practice of optimizing the LoRA factors (A, B) with Euclidean optimizers such as SGD (Robbins & Monro, 1951), Adam (Kingma & Ba, 2014), Adagrad (Duchi et al., 2011), RMSProp (Tieleman, 2012), etc. that misaligned with the geometry of the low-rank constraint. The same update ΔW can be represented by infinitely many factorizations: for any $A \in \mathbb{R}^{m\times r}$, $B \in \mathbb{R}^{n\times r}$ and any invertible matrix $S \in \mathbb{R}^{r\times r}$, we may write:

$$\Delta W = AB^\top = \widetilde{A}\widetilde{B}^\top, \quad \text{where} \quad \widetilde{A} = AS, \quad \widetilde{B} = BS^{-\top}. \tag{1}$$

Ideally, training should be *reparameterization (transformation) invariant*: the update to ΔW must not depend on which factorization (A, B) is used (Yen et al., 2024). Empirically, this lack of invariance manifests as unbalanced learning where one factor dominates and the other stalls, fragile hyperparameter sensitivity, and path-dependent solutions. These issues have prompted geometry-aware formulations. Riemannian treatments of low-rank models operate on the *fixed-rank manifold* rather than the ambient factor space, projecting gradients to the tangent space and retracting back to the manifold. Such steps can be implemented efficiently when $r \ll \min\{m, n\}$ and avoid forming

full-size matrices. Within the LoRA literature, existing Riemannian approaches either use standard SGD-type optimizers Mo et al. (2025, LORO) or rely on the Adam (Zhang & Pilanci, 2024) optimizer for auxiliary matrices within the chosen parameterization, which deviate them from the Riemannian framework and introduce dependence on parameterization.

In this work, we introduce a *fully Riemannian* framework for training LoRA that optimizes the adapter $X = \Delta W$ directly on the fixed-rank matrix manifold

$$\mathcal{M}_r \;=\; \{X \in \mathbb{R}^{m\times n} : \operatorname{rank}(X) = r\},$$

eliminating factorization ambiguity by construction. Central to our approach is ***Riemannion***, a new Riemannian optimizer on $\mathcal{M}_r$ that *generalizes the recently proposed Muon optimizer (Jordan et al., 2024)* to the fixed-rank setting. In contrast to prior Riemannian LoRA variants that port Adam-like mechanics to the manifold with ad hoc choices, our design inherits Muon's geometry-aligned normalization, yielding transformation invariance of the learned update. We further propose a Riemannian gradient–informed initialization that places the initial adapter at a good location on $\mathcal{M}_r$, and we provide a practical, low-overhead implementation that assembles projections, retractions, and vector transports via automatic differentiation, also following best practices from numerical linear algebra. Extensive experiments on LLM and diffusion architectures show consistent gains in convergence speed and final task performance over standard LoRA and recent state-of-the-art modifications.

Our contributions are as follows:

- **Riemannion: Muon on the fixed-rank manifold.** We derive *Riemannion*, the first optimizer that *generalizes Muon* to the manifold $\mathcal{M}_r$ of fixed rank matrices.
- **Riemannian gradient-informed initialization.** We propose an initialization strategy which yields best alignment between the initial Riemannian gradient and the Euclidean gradient. We also propose an efficient way for this strategy by using a randomized SVD algorithm with implicit matrix multiplication (Section 5). Finally, we show the connection of this initialization to LoRA-GA.
- **Efficient implementation with automatic differentiation.** We pay special attention to numerical implementation to make the method robust without any prominent overhead compared to vanilla LoRA at small ranks.
- **Comprehensive empirical validation.** We showcase the performance of our framework for fine-tuning LLMs and in subject-driven generation using diffusion models. Among positive effects that we observe are: boost in target metrics, improved convergence, and reduction of variance.

2 Related Work

The problem of an optimal initial guess selection for low-rank LLM adaptation has been addressed in a sequence of works: the authors Meng et al. (2024, PiSSA) have suggested a heuristic that involves using a low-rank truncated SVD of pretrained parameters as an initial point for LoRA and its orthogonal complement as frozen layer's parameters, so that the tuning process starts without changing the starting value of the loss function. A similar approach was implemented by Wang et al. (2025, MiLoRA) with the main difference of optimizing the smallest singular components of unadapted parameter matrix. A context-aware initialization was considered in (Yang et al., 2024, CorDA) and (Parkina & Rakhuba, 2025, COALA) proposes a numerically robust inversion-free framework for low-rank weighted approximations for this setting. Another idea is to initialize LoRA with a subset of left and right singular vectors of a doubled-rank truncated SVD of the loss function gradient at the starting parameters, proposed by Wang et al. (2024, LoRA-GA). We show direct connection of this method to our Riemannian initialization strategy and propose how to additionally significantly accelerate the computation of SVD using our approach. Attempting to overcome the asymmetry in the initialization of vanilla LoRA fine-tuning process, (Hayou et al., 2024, LoRA+) introduced a scale-free step size selection for LoRA factors.

Riemannian optimization is widely used for algorithms on matrix manifolds and allows for exploiting task geometry or imposing additional constraints. For example, a Riemannian solution for the extreme eigenpairs search problem was described in Absil et al. (2009); Baker (2008), a matrix

completion task, which is common in collaborative filtering for recommender systems, via optimization on the fixed-rank manifold (Vandereycken, 2013), a Riemannian approach on the manifold of matrices with orthonormal columns (the Stiefel manifold) was used by Wisdom et al. (2016) for diminishing the problem of vanishing and exploding gradients in recurrent neural networks, etc. The book Trendafilov & Gallo (2021) also presents a comprehensive description of useful manifolds for solutions of the data science problems. For the deeper understanding of applied differential geometry techniques see the books Absil et al. (2009) and Boumal (2023).

The idea of using Riemannian optimization has recently started to emerge for the large language models. For example, the fine-tuning of LLMs with the help of the Stiefel manifold was considered in the work Hu et al. (2024). The authors of (Zhang & Pilanci, 2024) introduced the Riemannian inspired modification of Adam. The authors of Mo et al. (2025, LORO) applied the Riemannian optimization techniques for pretraining LLMs on the fixed-rank manifold. Parametrization that is used in our work can potentially help in this setting as well, by additionally avoiding potential overheads and instabilities, arising due the explicit inversion of Gram matrices.

3 PRELIMINARIES

3.1 MUON OPTIMIZER

Muon is an optimizer designed specifically for matrix-valued parameters in a network's hidden layers. Empirically, it accelerates training on language and vision workloads while leaving scalar/vector parameters and the input/output layers to a conventional optimizer such as AdamW. At a high level, Muon takes the step that stochastic gradient descent with momentum (SGDM) would make on a weight matrix and *orthogonalizes* that update before applying it. Orthogonalization acts as a per-layer, per-step preconditioner that equalizes singular values of the update, which mitigates the collapse of updates into a few dominant directions (Jordan et al., 2024). More specifically, let $W \in \mathbb{R}^{n\times m}$ be a hidden-layer weight. With gradient $G_t = \nabla_W \mathcal{L}(W_t)$ and momentum $M_t = \beta M_{t-1} + G_t$, Muon computes

$$\widetilde{M}_t \approx \mathrm{Ortho}(M_t) \quad \text{and} \quad W_{t+1} = W_t - \eta\, \widetilde{M}_t,$$

where $\mathrm{Ortho}(\cdot)$ denotes the nearest semi-orthogonal matrix in Frobenius norm, i.e.,

$$\mathrm{Ortho}(G) = \arg\max_{O} \big\{ \|O - G\|_F : O^\top O = I \text{ or } OO^\top = I \big\}. \tag{2}$$

Computing $\mathrm{Ortho}(G)$ exactly amounts to taking the SVD $G = USV^\top$ and returning $UV^\top$, which is too slow to do at every iteration. Muon instead applies a Newton–Schulz (NS) iteration that—after normalizing G — implements a composition of a fixed low-degree polynomial in $GG^\top$ acting on G and converges to $UV^\top$. Leveraging efficient matrix multiplication operations results in a highly performant iteration. We will write $\widetilde{M}_t = NS(M_t)$ for short.

LMO interpretation. Muon's step admits a clean linear minimization oracle (LMO) interpretation (Bernstein, 2025). Indeed, consider the operator-norm unit ball $\mathcal{B}_2 = \{X : \|X\|_2 \le 1\}$. The linear minimization oracle (LMO) over $\mathcal{B}_2$ at matrix M_t given by its SVD $M_t = USV^\top$ is

$$UV^\top \in \operatorname*{Arg\,max}_{\|S\|_2 \le 1} \langle M_t, S \rangle. \tag{3}$$

Applying Muon to LoRA. In LoRA, a frozen weight $W_0 \in \mathbb{R}^{n\times m}$ is adapted via a low-rank update $W = W_0 + \alpha\, BA$ with $B \in \mathbb{R}^{n\times r}$, $A \in \mathbb{R}^{r\times m}$ and small r. Each trainable factor (A and B) is a 2D parameter, so Muon can be applied *per factor*:

$$M_t^{(A)} \leftarrow \beta M_{t-1}^{(A)} + \nabla_A \mathcal{L}, \quad A_{t+1} \leftarrow A_t - \eta_A\, \mathrm{NS}\big(M_t^{(A)}\big),$$
$$M_t^{(B)} \leftarrow \beta M_{t-1}^{(B)} + \nabla_B \mathcal{L}, \quad B_{t+1} \leftarrow B_t - \eta_B\, \mathrm{NS}\big(M_t^{(B)}\big).$$

Note that acting on the two factors separately makes Muon non–reparameterization-invariant: its per-factor orthogonalization depends on arbitrary scalings or rotations, skewing the weight-space step and often letting one factor dominate.

3.2 RIEMANNIAN OPTIMIZATION

Let $\mathcal{M}_r = \{X \in \mathbb{R}^{m\times n} : \operatorname{rank}(X) = r\} \subseteq \mathbb{R}^{m\times n}$ be a smooth manifold of fixed-rank matrices (Lee, 2003, Example 8.14). Let every point X of $\mathcal{M}_r$ be equipped with a *tangent plane* $\mathcal{T}_X\mathcal{M}_r$. Thinking geometrically, the tangent plane plays the role of the best local, flat approximation to this curved set: if you "zoom in" at X, the manifold looks like a plane. We will now discuss how to numerically parametrize points on a manifold and its tangent plane. Every rank-r matrix $X \in \mathcal{M}_r$ can be represented using matrices $A_L \in \mathbb{R}^{m\times r}, B_r \in \mathbb{R}^{n\times r}$ with orthonormal columns and a square matrix $G \in \mathbb{R}^{r\times r}$ as

$$X = A_L G B_R^\top. \tag{4}$$

For example, one may think of a thin SVD, in which case G becomes the diagonal matrix of singular values, but other representations are also possible and will be convenient for our purposes. Using (4), any tangent vector $\xi \in \mathcal{T}_X\mathcal{M}_r$ can be represented in a matrix factorization format:

$$\xi = \begin{bmatrix}\dot{A} & A_L\end{bmatrix}\begin{bmatrix}B_R & \dot{B}\end{bmatrix}^\top, \quad \dot{A}^\top A_L = 0, \tag{5}$$

so that any tangent vector can be identified in terms of the tuple: $(\dot{A}, \dot{B})$. Since the factor matrices contain $2r$ columns, we immediately have that $\operatorname{rank}\xi \leq 2r$. Another remarkable fact is that the point $X \in \mathcal{M}_r$ itself lies in the $\mathcal{T}_X\mathcal{M}_r$ with $\dot{A} = 0, \dot{B} = B$. Given a matrix $Z \in \mathbb{R}^{m\times n}$, its orthogonal projection $P_{\mathcal{T}_X\mathcal{M}_r} Z$ onto the tangent space $\mathcal{T}_X\mathcal{M}_r$ (with the parameterization given in (5)) can be computed as follows:

$$P_{T_X\mathcal{M}_r}(Z) = A_L A_L^\top Z \; + \; \left(I - A_L A_L^\top\right) Z B_R B_R^\top. \tag{6}$$

and, hence, can be represented in the form of (5) with $\dot{A} = (I - A_L A_L^\top) Z B_R$, and $\dot{B} = Z^\top A_L$.

Let $F\colon \mathbb{R}^{m\times n} \to \mathbb{R}$ be a differentiable function with the Euclidean gradient $\nabla F \in \mathbb{R}^{m\times n}$. Within the Riemannian optimization framework, we solve the following task optimization problem:

$$\min_{X\in\mathcal{M}_r} F(X).$$

When constructing the algorithms, we need to work with three key objects: Riemannian gradient, retraction and vector transport. The Euclidean gradient is a direction of the steepest local increase F. Therefore, it is common to use the Riemannian gradient — the direction of the steepest local increase of corresponding smooth function value along the manifold, which lies in the tangent space (Absil et al., 2009, chap. 3.6). Given the Euclidean gradient ∇F, one may endow the tangent space $\mathcal{T}_X\mathcal{M}_r$ with a natural scalar product and derive a formula for the direction of the local steepest ascent of F with respect to the manifold. This unique direction is called the Riemannian gradient and can be computed as follows:

$$\operatorname{grad} F(X) = P_{\mathcal{T}_X\mathcal{M}_r}\left(\nabla F(X)\right), \quad X \in \mathcal{M}_r. \tag{7}$$

A simple and robust retraction that maps a tangent step ξ (for example, ξ is a negative Riemannian gradient) back to the manifold is the truncated SVD:

$$R_X(\xi) \equiv R(X + \xi) = \operatorname{SVD}_r(X + \xi), \tag{8}$$

i.e., the best rank-r approximation of $X + \xi$ in Frobenius norm. Note that here we do not need to compute the full SVD and can utilize low-rank structure of X and ξ, leading to $\mathcal{O}((m+n)r^2 + r^3)$ operations (Absil & Oseledets, 2015). Finally, because tangent spaces change from an optimization step to step, momentum (an accumulated tangent vector) must be moved between them via a *vector transport*. For embedded manifolds like $\mathcal{M}_r$, a standard choice is the *projection transport*

$$\mathcal{T}_{Y\to X}(\xi) \; = \; P_{T_X\mathcal{M}_r}(\xi), \qquad \xi \in \mathcal{T}_Y\mathcal{M}_r, \tag{9}$$

which simply reprojects the same ambient matrix ξ onto the new tangent space at X.

4 RIEMANNION

In this section, we focus on the setting of parameter-efficient fine-tuning and, hence, to the fixed-rank manifold. For fine-tuning of one layer the optimization problem becomes:

$$\mathcal{L}(W + \Delta W) \to \min_{\Delta W\in\mathcal{M}_r},$$

where $\mathcal{L}$ is a differentiable loss function. Note that optimizing *on* $\mathcal{M}_r$ removes the ambiguity of factorized parameterizations, because all computations are carried out in the intrinsic space of the product X rather than in any particular factorization. So the formulas we write below will naturally be reparameterization-invariant. Let $G_t = P_{T_W\mathcal{M}_r}(\nabla\mathcal{L}(W_t))$ be the Riemannian gradient. A Riemannian *heavy-ball* (Polyak, 1964) momentum step reads

$$M_t \;=\; \beta\,\widehat{M}_{t-1} \;+\; G_t, \quad \widehat{M}_{t-1} \;=\; \mathcal{T}_{W_{t-1}\to W_t}(M_{t-1}), \tag{10}$$

$$\Delta W_{t+1} \;=\; R\left(\Delta W_t - \eta\, M_t\right), \tag{11}$$

with $M_0 = 0$, momentum parameter $\beta \in [0, 1)$, and stepsize $\eta > 0$.

Let us now discuss how to introduce a Muon-like variant of this iteration, which we refer to as Riemannion. A direct projection onto the set of orthogonal matrices $\mathrm{Ortho}(M_t)$ presents two challenges. First, such a step does not respect the underlying Riemannian geometry. To address this, we propose to find the best approximation of $\mathrm{Ortho}(M_t)$ on the tangent plane $T_W\mathcal{M}_r$. Such a solution is given via the projection onto the tangent plane $P_{T_W\mathcal{M}_r}(\mathrm{Ortho}(M_t))$. However, a second issue arises: although this projection is low-rank, its computation remains inefficient because the input matrix is of full rank. Let us notice that $M_t \in T_{\Delta W_t}\mathcal{M}_r$ and, hence, is of rank at most $2r$ (Section 3.2). At the same time, the LMO interpretation (Section 3.1) provides several admissible low-rank solutions, including one where $\mathrm{Orthor}(\cdot)$ replaces only the first $2r$ singular values with 1, while all others are set to 0. Consequently, we obtain the following update rule:

$$\widetilde{M}_t = P_{T_{\Delta W_t}\mathcal{M}_r}(\mathrm{Ortho}_r(M_t)). \tag{12}$$

Note that $\mathrm{Ortho}_r(\cdot)$ exactly preserves the column and row spaces of $M_t \in T_{\Delta W_t}\mathcal{M}_r$. Consequently, although $P_{T_{\Delta W_t}\mathcal{M}_r}(\mathrm{Ortho}_r(M_t))$ does not yield singular values exactly equal to 1, in practice they remain in a close proximity. This behavior is reminiscent of the Newton–Schulz iteration, which likewise produces approximate singular values. To eliminate this inexactness and obtain an accurate solution in the intersection of $T_{\Delta W_t}$ with the set of matrices whose first $2r$ singular values are exactly 1, one could formally apply the alternating projection method (Cheney & Goldstein, 1959; Boyd & Dattorro, 2003, Theorem 4, Section 2):

$$\widetilde{M}_t = P_{T_{\Delta W_t}\mathcal{M}_r}(\mathrm{Ortho}_r(\ldots P_{T_{\Delta W_t}\mathcal{M}_r}(\mathrm{Ortho}_r(M_t)))).$$

However, such a procedure is more computationally excessive, and our experiments indicate that it has little to no impact on the overall convergence of the optimizer.

Algorithm 1 `OrthoLR` (efficient computation of $\mathrm{Ortho}_r(\xi)$ for $\xi \in T_X\mathcal{M}_r$).

Require: $\xi \in T_X\mathcal{M}_r$ given by $(\dot{A}, \dot{B})$ from (5); A_L, B_R such that $X = A_L G B_R^\top$ as in (4)
Ensure: $A \in \mathbb{R}^{m\times 2r}, B \in \mathbb{R}^{m\times 2r}$: $AB^T = \mathrm{Ortho}(\xi)$.

1: $Q_L, T_L = \mathtt{qr}([A_L, \dot{A}]), \quad Q_R, T_r = \mathtt{qr}([\dot{B}, B_R]^\top).$ // $\mathcal{O}\left((m+n)r^2\right)$
2: $U_L, \_, V_r^\top = \mathtt{SVD}\left(T_L\, T_R^\top\right).$ // $\mathcal{O}\left(r^3\right)$
3: $\dot{A} = Q_L U_L, \quad \dot{B} = Q_R V_R.$ // $\mathcal{O}\left((m+n)r^2\right)$

Algorithm 2 `ProjectLR` (efficient computation of $P_{T_X\mathcal{M}_r}(Z)$ for a rank-r' matrix Z).

Require: $A \in \mathbb{R}^{m\times r'}, B \in \mathbb{R}^{n\times r'}$ such that $Z = AB^\top$; A_L, B_R such that $X = A_L G B_R^\top$ as in (4).
Ensure: $\xi = P_{T_X\mathcal{M}_r}(Z)$ given by $(\dot{A}, \dot{B})$ from (5).

1: $\dot{A} \coloneqq (A - A_R(A_R^\top A))(B^\top B_R), \quad \dot{B} \coloneqq B(A^\top A_L)$ // $\mathcal{O}((m+n)r'^2)$

Let us finally show that $\widetilde{M}_t$ from (12) can be computed efficiently using $\mathcal{O}((m+n)r^2 + r^3)$ arithmetic operations. Indeed, first of all we need to apply $\mathrm{Ortho}_r(\cdot)$ to a tangent vector M_t. From (6), we know that a tangent vector can be represented in a form of a rank-$2r$ matrix. Such a representation can always be transformed into the compact SVD form with 2 QR decompositions and a single full SVD of a $2r \times 2r$ matrix, see Algorithm 1 called `OrthoLR`. As a next step, we need to project the obtained result (decomposed matrix of rank $2r$) onto the tangent plane. The operation can also be done efficiently via (6) and is summarized in Algorithm 2 called `ProjectLR`.

5 LOCALLY OPTIMAL INITIALIZATION (LOI)

Once the theoretical framework for the Riemannian optimizer is established, it is natural to consider an initialization scheme that accounts for the underlying Riemannian geometry. Given any $\Delta W \in \mathcal{M}_r$, we may write

$$\mathcal{L}(W) = \mathcal{L}(\underbrace{W - \Delta W}_{W'} + \Delta W) = \mathcal{L}\left(W' + \Delta W\right). \tag{13}$$

This raises the question: how should ΔW be chosen to ensure the fastest loss decrease along the manifold? The solution is to consider the following optimization task:

$$\Delta W_*^{(0)} \in \underset{\Delta W \in \mathcal{M}_r}{\operatorname{Arg\,max}} \|P_{\mathcal{T}_{\Delta W}\mathcal{M}_r} \nabla_W \mathcal{L}(W)\|_F^2. \tag{14}$$

Since $P_{\mathcal{T}_{\Delta W}\mathcal{M}_r}$ is an orthogonal projection matrix to the tangent plane, the task (14) essentially seeks for the point on the fixed-rank manifold, whose tangent space has most alignment with the Euclidean gradient. In other words, this means that the direction of the steepest local function decrease alongside the manifold is aligned with the full model tuning direction. The solution to this task is presented in Theorem 5.1.

Theorem 5.1. *Let the SVD of $\nabla_W \mathcal{L}(W)$ be:*

$$\nabla_W \mathcal{L}(W) = \begin{bmatrix} U_{1,r} & U_{r,2r} & U_\perp \end{bmatrix} \begin{bmatrix} \Sigma_{1,r} & 0 & 0 \\ 0 & \Sigma_{r,2r} & 0 \\ 0 & 0 & \Sigma_\perp \end{bmatrix} \begin{bmatrix} V_{1,r} & V_{r,2r} & V_\perp \end{bmatrix}^\top,$$

and let also $\sigma_{2r} \neq \sigma_{2r+1}$. Then any optimal solution $\Delta W_*^{(0)}$ to the problem (14) has the form:

$$\Delta W_*^{(0)} \in \left\{ [U_{1,r}, U_{r,2r}\Sigma_{r,2r}] \begin{bmatrix} S_{11} \\ S_{21} \end{bmatrix} \begin{bmatrix} C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} \Sigma_{1,r} V_{1,r}^\top \\ V_{r,2r} \end{bmatrix} \,\middle|\, S = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \in \mathrm{GL}_{2r}(\mathbb{R}),\ S^{-1} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \right\}. \tag{15}$$

Proof. See Appendix A. □

In our experiments, we use $S = \begin{bmatrix} \alpha I_r & 0 \\ 0 & I_r \end{bmatrix}$, obtaining: $\Delta W_*^{(0)} = \alpha U_{1,r} V_{r,2r}^\top \in \mathcal{M}_r, \alpha \in \mathbb{R} \setminus \{0\}$.

Interestingly, Theorem 5.1 relates to the findings of Wang et al. (2024), although their analysis neither adopts a Riemannian framework nor addresses parametrization-free optimization. Our optimizer further differs from Zhang & Pilanci (2024) and Mo et al. (2025, LORO) in that it avoids inversion of the Gram matrix. As a result, the method remains stable as $\|\Delta W_*^{(0)}\| \to 0$. Empirical results indicate that initializing $\Delta W_*^{(0)}$ with a small norm leads to improved performance. The procedure for selecting the scaling parameter α is described in Appendix E.

6 SINGLE BACKWARD-PASS GRADIENT TRICK

This section introduces an efficient method for computing gradient-times–matrix in a matrix-free way, at a computational cost equivalent to a *single backward pass*. We then apply this technique in both the LOI initialization procedure in Algorithm 3 and the Riemannion optimizer in Algorithm 4.

The calculation of the full fine-tuning loss gradient $\nabla_W \mathcal{L}(W)$ with pretrained parameters $W \in \mathbb{R}^{m \times n}$ is computationally expensive. At the same time, for our framework we only need to compute the products $(\nabla_W \mathcal{L}(W)^\top M)$ or $(\nabla_W \mathcal{L}(W) N)$ for some $M \in \mathbb{R}^{m \times r}, N \in \mathbb{R}^{n \times r}$. Using the trick from (Novikov et al., 2022), we may calculate both quantities simultaneously using a *single forward-backward pass* with a doubled rank representation.

First, we initialize differentiable parameters $Z_1 = 0 \in \mathbb{R}^{m \times r}$ and $Z_2 = 0 \in \mathbb{R}^{n \times r}$ and perform a simple forward pass $L := \mathcal{L}\left(W + Z_1 N^\top + M Z_2^\top\right)$. This step does not violate the pipelines of LoRA framework since it is equivalent to a standard LoRA forward pass with special adapter:

$$Z_1 N^\top + M Z_2^\top = \begin{bmatrix} Z_1 & M \end{bmatrix} \begin{bmatrix} N & Z_2 \end{bmatrix}^\top = \tilde{A}\tilde{B}^\top, \quad \tilde{A} \in \mathbb{R}^{m \times 2r}, \tilde{B} \in \mathbb{R}^{n \times 2r}.$$

Then we invoke an autodiff algorithm for value L, which ensures us both:

$$\begin{aligned}\nabla_{Z_1} L &= \nabla_{Z_1} \mathcal{L}\left(W + Z_1 N^\top + M Z_2^\top\right)|_{Z_1=0, Z_2=0} = \nabla_W \mathcal{L}(W) N, \\ \nabla_{Z_2} L &= \nabla_{Z_2} \mathcal{L}\left(W + Z_1 N^\top + M Z_2^\top\right)|_{Z_1=0, Z_2=0} = \nabla_W \mathcal{L}(W)^\top M.\end{aligned} \tag{16}$$

Note, that if $\Delta W = A_L B^\top = A B_R^\top$ from $W + \Delta W$, then one may take $N = B_R, M = A_L, Z_1 = 0, Z_2 = B, Y = W + \Delta W$ and the exact same operations work. Notably, (16) is crucial for computation of the Riemannian gradient in (6). Therefore the proposed approach is a key building block that allows us to avoid forming the full fine-tune gradient loss and effectively compute matrix-matrix multiplications. This idea is the core for both: LOI initialization (Algorithm 3) and Riemannion optimizer (Algorithm 4).

Randomized SVD for efficient initialization The computation of the $2r$-truncated SVD of the full loss gradient, as required by Theorem 5.1, has asymptotic complexity $\mathcal{O}(\min\{m, n\}mn)$, which may be infeasible for large-scale models. In addition, explicitly forming this gradient is itself computationally expensive, and thus we need to avoid it in practice. To overcome this problem, we propose to use a randomized SVD with power iterations (see (Halko et al., 2011)), which we also enhance with our *one-step gradient trick* as is described in the Algorithm 3. In a nutshell, we need to compute $\left(\nabla_W \mathcal{L}(W) \nabla_W \mathcal{L}(W)^\top\right)^q Y$, where $Y = \nabla_W \mathcal{L}(W)\Omega$ and Ω is sampled from standard normal distribution. This iteration can be done in a robust manner using QR decompositions. The steps $2, 4, 5, 6$ correspond to a trick from (16).

Overall, the LOI search procedure has asymptotic complexity $\mathcal{O}((m+n)r^2)$, plus $2(q+1)$ additional backward passes. Moreover, since LOI search is executed only once before fine-tuning (which typically involves thousands of backward passes), its runtime overhead is negligible, taking merely 0.25% of the total fine-tuning wall-clock time in our experiments. Section 7.1, about 2 thousand optimization steps were carried out and Algorithm 3 accounted for merely 0.25% of the total fine-tuning wall-clock time.

Algorithm 3 `BackPropRSVD`

Require: Weights $W \in \mathbb{R}^{m \times n}$, rank $r \in \mathbb{N}$, oversampling parameter p, power-step parameter q.
Ensure: Randomized r-truncated SVD (U_r, Σ_r, V_r) of $\nabla_W \mathcal{L}(W)$.

1: Choose $k = r + p$, Sample $\Omega \in \mathbb{R}^{n \times k} \sim \mathcal{N}(0, 1)$. // $\mathcal{O}(nr)$
2: $Y := \mathtt{qr}(\nabla_A \mathcal{L}(W + A\Omega^\top)|_{A=0}).\mathtt{Q}$. // 1 backward pass $+\mathcal{O}\left(mr^2\right)$
3: **for** $i := 1, \ldots, q$ **do**
4: $\quad Y := \mathtt{qr}([\nabla_B \mathcal{L}(W + YB^\top)|_{B=0}]^\top).\mathtt{Q}$. // 1 backward pass $+\mathcal{O}\left(nr^2\right)$
5: $\quad Y := \mathtt{qr}(\nabla_A \mathcal{L}(W + AY^\top)|_{A=0}).\mathtt{Q}$. // 1 backward pass$+\mathcal{O}\left(mr^2\right)$
6: $Y := [\nabla_B \mathcal{L}(W + YB^\top)|_{B=0}]^\top$. // 1 backward pass
7: $U, \Sigma, V^\top := \mathtt{truncSVD}\,(Y, r)$. // $\mathcal{O}\left(nr^2\right)$
8: $YU, \Sigma, V^\top$ // $\mathcal{O}\left(mr^2\right)$

Efficient Riemannion implementation The Riemannion optimizer is presented in Algorithm 4. Similar to vanilla Muon, it relies on a chosen `Ortho` procedure. Since the LoRA approach represents the fine-tuning shift ΔW with low-rank matrices, we employ an SVD-based `Ortho` procedure. For computational efficiency, this procedure is adapted to the fixed-rank manifold, as described in Section 4 and implemented in the `OrthoLR` (Algorithm 1).

In detail, in step 1 we calculate the Riemannian gradient components via a *single backward call* (16). Then, in step 2 the algorithm transports the Heavy-Ball tangent direction to the current point via Algorithm 2 that provides a simple but effective implementation of (6). In line 4, we compute the final optimization direction on the tangent space of the given point with a Heavy-Ball momentum coefficient β. In line 4-5 algorithm performs the Riemannion projection procedure and the retraction iteration. The algorithm finalizes with saving the obtained minimization direction for momentum in line 6 and calculating of a new point representation in step 7.

Overall, one iteration of the Riemannion loop has asymptotic complexity $\mathcal{O}((m + n)r^2 + r^3)$ and additionally the same number of backward passes as vanilla LoRA. For comparison, the Euclidean Muon optimizer for LoRA (Section 3.1) exhibits the same asymptotic complexity. The complete framework and its time performance are summarized in Algorithm 5 and Appendix H.

Algorithm 4 `One step of Riemannion`

Require: Weight matrix $W' \in \mathbb{R}^{m\times n}$, rank $r \in \mathbb{N}$, initial point A_L, B_R, Heavy-ball momentum $A_{\rm HB}, B_{\rm HB}$, step size η, momentum coefficient β, weight decay coefficient γ.
Ensure: Tuning parameters $\Delta W^* \in \mathcal{M}_r$.

$\dot{A}, \dot{B} := \nabla_{Z_1}\mathcal{L}\left(W' + Z_1 B_R^\top + A_L Z_2^\top\right)|_{Z_1=0,Z_2=B}$,
$\qquad \nabla_{Z_2}\mathcal{L}\left(W' + Z_1 B_R^\top + A_L Z_2^\top\right)|_{Z_1=0,Z_2=B}$. // 1 backward pass
$\dot{A}_{\rm prev}, \dot{B}_{\rm prev} :=$ `ProjectLR` $((A_{\rm M}, B_{\rm M}), A_L, B_R)$. // $\mathcal{O}\left((m+n)r^2\right)$
$\dot{A}, \dot{B} := \beta\dot{A}_{\rm prev} + (I - A_L A_L^\top)\dot{A},\ \beta\dot{B}_{\rm prev} + \dot{B}$ // $\mathcal{O}\left((n+m)r^2\right)$
$\dot{A}, \dot{B} :=$ `ProjectLR`(`OrthoLR`$(A_L, B_R, \dot{A}, \dot{B}, r,), A_L, B_R)$ // $\mathcal{O}\left((m+n)r^2 + r^3\right)$
$U, \Sigma, V^\top :=$ `RetractionLR`$([-\eta\dot{A}, A_L], [B_R, -\eta(\dot{B} + \gamma B_R)])$ // $\mathcal{O}\left((m+n)r^2 + r^3\right)$
$A_{\rm HB}, B_{\rm HB} := [\dot{A}, A_L],\ [B_R, \dot{B}]$
$A_L, B := U, \Sigma V^\top$ // $\mathcal{O}\left(nr^2\right)$

7 EXPERIMENTS

In this section, we also employ the term LOI (Locally Optimal Initialization), referring to the proposed initialization scheme described in Section 5. This is done to distinguish between using RiemannLoRA as an optimizer with zero initialization (as in basic LoRA) and RiemannLoRA-LOI, which is the proposed combination of initialization and optimization. We conduct a series of LLM fine-tuning experiments for different tasks. All of the experiments were computed on NVIDIA V100-32Gb GPU and A100-80Gb GPU. We ran all the experiments within ~ 2000 GPU hours.

7.1 COMMONSENSE REASONING FINE-TUNING

The results were obtained for the benchmark (Clark et al. (2019, BoolQ), Bisk et al. (2020, PIQA), Sap et al. (2019, SIQA), Zellers et al. (2019, hellaswag), Sakaguchi et al. (2021, winogrande),Clark et al. (2018, ARC), Mihaylov et al. (2018, OBQA)) common reasoning. The structure of the dataset is described in Appendix D. In the following experiments we conduct a fine-tuning procedures for multilayer perceptron (MLP) and attention layers of Llama 3 8b model (Dubey et al. (2024)).

The commonsense reasoning tasks comprise of 8 sub-tasks, each of them contains a predefined training and a testing set. We follow the setting of Hu et al. (2023) and amalgamate the training datasets from all 8 tasks to create the final training dataset and conduct evaluations on the individual testing dataset for each task. The hyperparameter tuning protocol and the selected hyperparameters are provided in E. Table 1 contains the accuracy of the trained model's responses on the test dataset. Within the LoRA framework, the proposed Riemannian fine-tuning method delivers clear performance gains. The Riemannion optimizer consistently outperforms LoRA, DoRA, and achieves superior metric results compared to the standard Muon optimizer applied to LoRA factors (see Section 3.1). Furthermore, relative to other Riemannian-geometry–aware approaches such as RPrecAdamW (Zhang & Pilanci, 2024), our method also demonstrates better results. Finally, the variance of outcomes for the proposed method is the smallest among all compared approaches.

Table 1: The average accuracy (in %) among 8 tasks of fine-tuned Llama 3-8b using different approaches, tested on Commonsense Reasoning benchmark. LoRA rank is set to 16.

Task Initialization	BoolQ	PIQA	SIQA	hella- swag	wino- grande	ARC-E	ARC-C	OBQA	All
Raw	65.0	76.6	73.0	66.1	61.3	92.5	82.3	79.6	74.5
Adam	$74.8_{\pm 1.9}$	$89.8_{\pm 0.9}$	$82.6_{\pm 0.6}$	$96.2_{\pm 0.3}$	$87.9_{\pm 1.2}$	$92.4_{\pm 0.7}$	$84.9_{\pm 0.7}$	$88.5_{\pm 0.4}$	$87.1_{\pm 0.6}$
DoRA	$74.8_{\pm 0.8}$	$89.4_{\pm 0.5}$	$82.4_{\pm 0.7}$	$95.9_{\pm 0.1}$	$87.8_{\pm 0.4}$	$90.7_{\pm 1.2}$	$83.8_{\pm 0.7}$	$87.8_{\pm 0.6}$	$86.6_{\pm 0.3}$
Muon	$72.9_{\pm 0.0}$	$86.4_{\pm 0.5}$	$80.8_{\pm 0.2}$	$94.1_{\pm 0.2}$	$84.4_{\pm 0.0}$	$84.2_{\pm 0.9}$	$77.3_{\pm 2.5}$	$83.9_{\pm 1.1}$	$83.0_{\pm 0.6}$
DoneRITE	$72.2_{\pm 0.3}$	$88.6_{\pm 0.1}$	$82.0_{\pm 0.6}$	$95.1_{\pm 0.1}$	$85.6_{\pm 0.2}$	$87.7_{\pm 1.5}$	$79.3_{\pm 2.6}$	$85.7_{\pm 0.5}$	$84.5_{\pm 0.5}$
RPrecAdamW	$\mathbf{75.8}_{\pm 0.4}$	$89.5_{\pm 0.4}$	$82.4_{\pm 0.2}$	$96.1_{\pm 0.2}$	$87.7_{\pm 0.9}$	$90.6_{\pm 1.6}$	$84.1_{\pm 1.1}$	$87.7_{\pm 0.5}$	$86.8_{\pm 0.4}$
Riemannion	$75.7_{\pm 0.7}$	$\mathbf{91.2}_{\pm \mathbf{0.2}}$	$\mathbf{83.5}_{\pm \mathbf{0.6}}$	$\mathbf{96.7}_{\pm 0.0}$	$\mathbf{88.6}_{\pm 0.4}$	$\mathbf{93.6}_{\pm 0.3}$	$\mathbf{86.4}_{\pm \mathbf{0.4}}$	$\mathbf{89.3}_{\pm 0.8}$	$\mathbf{88.1}_{\pm 0.2}$

Figure 2: Visual results for Subject-driven generation on 600 training step.

7.2 Subject-driven generation

Subject-driven generation (Ruiz et al., 2023; Gal et al., 2023) is a task in which the user provides several reference photos of an object, called a concept, and uses a diffusion model to generate this concept with certain conditions (e.g., a textual prompt). One way to solve this task is to fine-tune a pre-trained diffusion model using this small set of reference images. However, this technique leads to a degradation in understanding of the conditions and to a fast overfitting of the concept. Furthermore, it requires a high computational cost due to the large number of trainable parameters. This is why previous works, such as (Qiu et al., 2023; Liu et al., 2024; Hu et al., 2022; Tewel et al., 2023; Han et al., 2023; Gorbunov et al., 2024), propose training only a lightweight parameterization for the base model. In this section, we demonstrate the performance of our parameterization in this task.

In our experiments, we used Stable Diffusion 2 (Rombach et al., 2022) as the base model. We choose LoRA Hu et al. (2022) as a baseline and train both models with ranks of 4, 8 and 16. We predict the parameterization of the `q`, `k`, `v`, and `out.0` matrices in all attention layers. The Dreambooth dataset (Ruiz et al., 2023) was used in all our experiments. LoRA was trained using the Adam optimizer. In Figure 2, we present a visual comparison of LoRA with different ranks. As can be seen, even for complex concepts such as 'robot toy', our method requires only 600 steps to learn the concept while preserving appropriate text similarity. We found that for subject-driven generation tasks, the lower the rank, the faster our method converges. To evaluate this, we also calculated metrics for different learning rates on a subset of the Dreambooth dataset. We use CLIP to measure text similarity and DINO to measure image similarity. Figure 1 shows that, even with different learning rates, our method achieves more accurate results in concept preservation. Further details and a visual comparison can be found in the Appendix F.

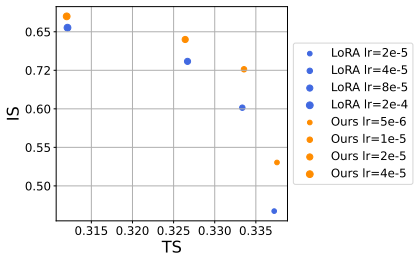

Figure 1: Comparison of text and image similarities for LoRA and our method with rank 4 at different learning rates on 400 step.

8 Conclusion

In this work, we propose a novel fully Riemannian framework that integrates a new muon-based optimization method, locally-optimal initialization, and an efficient implementation. This integrated approach yields a reliable reparametrization-invariant method that outperforms competing approaches on fine-tuning large language models (LLMs) and exhibits additional favorable properties for low-rank approximations in diffusion models. Given the promising empirical results, a natural direction for future research is to investigate the theoretical properties of the proposed method.

9 Reproducibility Statement

The hyperparameter selection procedure is described in Appendix E. The datasets used in the experiments and the corresponding preprocessing steps are detailed in Appendix D. The proof of the stated assumptions is provided in Appendix A. The hyperparameters for Subject-driven generation task are provided in the Appendix F.

References

P-A Absil and Ivan V Oseledets. Low-rank retractions: a survey and new results. *Computational Optimization and Applications*, 62(1):5–29, 2015.

P-A Absil, Robert Mahony, and Rodolphe Sepulchre. Optimization algorithms on matrix manifolds. In *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2009.

Christopher G Baker. *Riemannian manifold trust-region methods with applications to eigenproblems*. The Florida State University, 2008.

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

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. PIQA: Reasoning about Physical Commonsense in Natural Language. In *Proceedings of the AAAI conference on artificial intelligence*, number 05 in 34, pp. 7432–7439, 2020.

Nicolas Boumal. *An introduction to optimization on smooth manifolds*. Cambridge University Press, 2023.

Stephen Boyd and Jon Dattorro. Alternating projections. 2003. URL https://web.stanford.edu/class/ee392o/alt_proj.pdf.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Ward Cheney and Allen A Goldstein. Proximity maps for convex sets. *Proceedings of the American Mathematical Society*, 10(3):448–450, 1959.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 2924–2936. Association for Computational Linguistics, 2019. doi: 10.18653/V1/N19-1300. URL https://doi.org/10.18653/v1/n19-1300.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *CoRR*, abs/1803.05457, 2018. URL http://arxiv.org/abs/1803.05457.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan

Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. The Llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. doi: 10.48550/ARXIV.2407.21783. URL https://doi.org/10.48550/arXiv.2407.21783.

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

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit Haim Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=NAQvF08TcyG.

Mikhail Gorbunov, Nikolay Yudin, Vera Soboleva, Aibek Alanov, Alexey Naumov, and Maxim Rakhuba. Group and shuffle: Efficient structured orthogonal parametrization. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011. doi: 10.1137/090771806. URL https://doi.org/10.1137/090771806.

Ligong Han, Yinxiao Li, Han Zhang, Peyman Milanfar, Dimitris Metaxas, and Feng Yang. Svdiff: Compact parameter space for diffusion fine-tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7323–7334, 2023.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. LoRA+: Efficient Low Rank Adaptation of Large Models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=NEv8YqBROO.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-Rank Adaptation of Large Language Models. *ICLR*, 1(2):3, 2022.

Jiang Hu, Jiaxi Cui, Lin Lin, Zaiwen Wen, Quanzheng Li, et al. Retraction-free optimization over the Stiefel manifold with application to the loRA fine-tuning, 2024.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 5254–5276. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.319. URL https://doi.org/10.18653/v1/2023.emnlp-main.319.

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

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

John M Lee. *Smooth manifolds*. Springer, 2003.

Weiyang Liu, Zeju Qiu, Yao Feng, Yuliang Xiu, Yuxuan Xue, Longhui Yu, Haiwen Feng, Zhen Liu, Juyeon Heo, Songyou Peng, Yandong Wen, Michael J. Black, Adrian Weller, and Bernhard Schölkopf. Parameter-efficient orthogonal finetuning via butterfly factorization. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=7NzgkEdGyr.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal Singular Values and Singular Vectors Adaptation of Large Language Models. *Advances in Neural Information Processing Systems*, 37:121038–121072, 2024.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? A new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 2381–2391. Association for Computational Linguistics, 2018. doi: 10.18653/V1/D18-1260. URL https://doi.org/10.18653/v1/d18-1260.

Zhanfeng Mo, Long-Kai Huang, and Sinno Jialin Pan. Parameter and Memory Efficient Pretraining via Low-rank Riemannian Optimization. In *The Thirteenth International Conference on Learning Representations*, 2025.

Alexander Novikov, Maxim Rakhuba, and Ivan Oseledets. Automatic differentiation for Riemannian optimization on low-rank matrix and tensor-train manifolds. *SIAM Journal on Scientific Computing*, 44(2):A843–A869, 2022.

Uliana Parkina and Maxim Rakhuba. Coala: Numerically stable and efficient framework for context-aware low-rank approximation, 2025. URL https://arxiv.org/abs/2507.07580.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

Zeju Qiu, Weiyang Liu, Haiwen Feng, Yuxuan Xue, Yao Feng, Zhen Liu, Dan Zhang, Adrian Weller, and Bernhard Schölkopf. Controlling text-to-image diffusion by orthogonal finetuning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=K30wTdIIYc.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, June 2022.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22500–22510, 2023.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. *Communications of the ACM*, 64(9):99–106, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. SocialIQa: Commonsense Reasoning about Social Interactions. *CoRR*, abs/1904.09728, 2019. URL http://arxiv.org/abs/1904.09728.

Yoad Tewel, Rinon Gal, Gal Chechik, and Yuval Atzmon. Key-locked rank one editing for text-to-image personalization. In *ACM SIGGRAPH 2023 Conference Proceedings*, pp. 1–11, 2023.

Tijmen Tieleman. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26, 2012.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023a. doi: 10.48550/ARXIV.2302.13971. URL https://doi.org/10.48550/arXiv.2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023b. doi: 10.48550/ARXIV.2307.09288. URL https://doi.org/10.48550/arXiv.2307.09288.

Nickolay Trendafilov and Michele Gallo. *Multivariate data analysis on matrix manifolds*. Springer, 2021.

Bart Vandereycken. Low-rank matrix completion by Riemannian optimization. *SIAM Journal on Optimization*, 23(2):1214–1236, 2013.

Hanqing Wang, Yixia Li, Shuo Wang, Guanhua Chen, and Yun Chen. MiLoRA: Harnessing minor singular components for parameter-efficient LLM finetuning. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2025 - Volume 1: Long Papers, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pp. 4823–4836. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.NAACL-LONG.248. URL https://doi.org/10.18653/v1/2025.naacl-long.248.

Shaowen Wang, Linxi Yu, and Jian Li. LoRA-GA: Low-rank adaptation with gradient approximation. *Advances in Neural Information Processing Systems*, 37:54905–54931, 2024.

Scott Wisdom, Thomas Powers, John Hershey, Jonathan Le Roux, and Les Atlas. Full-Capacity Unitary Recurrent Neural Networks. *Advances in neural information processing systems*, 29, 2016.

Yibo Yang, Xiaojie Li, Zhongzhu Zhou, Shuaiwen Song, Jianlong Wu, Liqiang Nie, and Bernard Ghanem. CorDA: Context-oriented decomposition adaptation of large language models for task-aware parameter-efficient fine-tuning. *Advances in Neural Information Processing Systems*, 37: 71768–71791, 2024.

Jui-Nan Yen, Si Si, Zhao Meng, Felix Yu, Sai Surya Duvvuri, Inderjit S Dhillon, Cho-Jui Hsieh, and Sanjiv Kumar. Lora done rite: Robust invariant transformation equilibration for lora optimization. *arXiv preprint arXiv:2410.20625*, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/V1/P19-1472. URL https://doi.org/10.18653/v1/p19-1472.

Fangzhao Zhang and Mert Pilanci. Riemannian preconditioned loRA for fine-tuning foundation models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

A INITAL POINT SEARCH

In this section we introduce the proof of Theorem 5.1

Proof. In order to represent the optimization task (14) in a more simple way, we will use a slightly different formula for the projection of the full loss gradient onto the tangent space of the fixed-rank manifold. Let

$$\Delta W = A_L B^\top = A B_R^\top \in \mathcal{M}_r,$$

and the tangent space is parametrized as follows

$$\mathcal{T}_{\Delta W}\mathcal{M}_r = \{\dot{A}B_R^\top + A_L\dot{B}^\top \mid \dot{B} \in \mathbb{R}^{n\times r}, \dot{A} \in \mathbb{R}^{m\times r}, A_L^\top \dot{A} = 0\},$$

then one may derive an orthogonal projection formula for any matrix Z (see, e.g. (Boumal, 2023, eq. 7.53))

$$P_{\mathcal{T}_{\Delta W}\mathcal{M}_r} Z = Z - (I - A_L A_L^\top) Z (I - B_R B_R^\top) \in \mathcal{T}_{\Delta W}\mathcal{M}_r.$$

As the tangent space is a linear space, the operation of orthogonal projection can be written as an optimization task

$$P_{\mathcal{T}_{\Delta W}\mathcal{M}_r} Z = \underset{\xi \in \mathcal{T}_{\Delta W}\mathcal{M}_r}{\arg\min} \|Z - \xi\|_F^2. \tag{17}$$

Since

$$\begin{aligned}
\|\nabla\mathcal{L}\|_F^2 &= \| \left(\nabla\mathcal{L} - P_{\mathcal{T}_{\Delta W}\mathcal{M}_r}\nabla\mathcal{L}\right) + P_{\mathcal{T}_{\Delta W}\mathcal{M}_r}\nabla\mathcal{L}\|_F^2 = \\
&= \|\nabla\mathcal{L} - P_{\mathcal{T}_{\Delta W}\mathcal{M}_r}\nabla\mathcal{L}\|_F^2 + \|P_{\mathcal{T}_{\Delta W}\mathcal{M}_r}\nabla\mathcal{L}\|_F^2 + 2\langle \nabla\mathcal{L} - P_{\mathcal{T}_{\Delta W}\mathcal{M}_r}\nabla\mathcal{L}, P_{\mathcal{T}_{\Delta W}\mathcal{M}_r}\nabla\mathcal{L}\rangle \\
&= \|\nabla\mathcal{L} - P_{\mathcal{T}_{\Delta W}\mathcal{M}_r}\nabla\mathcal{L}\|_F^2 + \|P_{\mathcal{T}_{\Delta W}\mathcal{M}_r}\nabla\mathcal{L}\|_F^2 + 2\langle P_{\mathcal{T}_{\Delta W}\mathcal{M}_r}\left(\nabla\mathcal{L} - P_{\mathcal{T}_{\Delta W}\mathcal{M}_r}\nabla\mathcal{L}\right), \nabla\mathcal{L}\rangle \\
&= \|\nabla\mathcal{L} - P_{\mathcal{T}_{\Delta W}\mathcal{M}_r}\nabla\mathcal{L}\|_F^2 + \|P_{\mathcal{T}_{\Delta W}\mathcal{M}_r}\nabla\mathcal{L}\|_F^2 + 2\langle \underbrace{P_{\mathcal{T}_{\Delta W}\mathcal{M}_r}\nabla\mathcal{L} - P_{\mathcal{T}_{\Delta W}\mathcal{M}_r}\nabla\mathcal{L}}_{=0}, \nabla\mathcal{L}\rangle \\
&= \|\nabla\mathcal{L} - P_{\mathcal{T}_{\Delta W}\mathcal{M}_r}\nabla\mathcal{L}\|_F^2 + \|P_{\mathcal{T}_{\Delta W}\mathcal{M}_r}\nabla\mathcal{L}\|_F^2
\end{aligned}$$

so the optimization task (14) is equivalent to

$$\|\nabla\mathcal{L} - P_{\mathcal{T}_{\Delta W}\mathcal{M}_r}\nabla\mathcal{L}\|_F^2 \to \min_{\Delta W},$$

which, in turn, using (17) is equivalent to the task

$$\min_{\Delta W} \min_{\xi \in \mathcal{T}_{\Delta W}\mathcal{M}_r} \|\nabla\mathcal{L} - \xi\|_F^2.$$

Due to the fact that every vector of the tangent space is an element of a set of all matrices with $2r$-bounded rank $\mathcal{M}_{\leq 2r}$, one may use the Eckart-Young-Mirsky theorem (see (Eckart & Young, 1936)) to obtain the following lower bound:

$$\|\nabla\mathcal{L} - P_{\mathcal{T}_{\Delta W}\mathcal{M}_r}\nabla\mathcal{L}\|_F^2 \geq \|\nabla\mathcal{L} - \mathrm{truncSVD}\left(\nabla\mathcal{L}, 2r\right)\|_F, \forall\, \Delta W \in \mathcal{M}_r, \xi \in \mathcal{T}_{\Delta W}\mathcal{M}_r. \tag{18}$$

But it is possible to ensure $\Delta W_*^{(0)} \in \mathcal{M}_r$ and $\xi_* \in \mathcal{T}_{\Delta W_*^{(0)}}\mathcal{M}_r$ which turn the inequality (18) into the equality. One may take

$$\begin{aligned}
&\Delta W_*^{(0)} = A_L(\alpha B)^\top = \alpha U_{1,r} V_{r,2r}^\top = (\alpha A) B_R^\top, \quad \alpha \in \mathbb{R}, \\
&\xi_* = \dot{A}B_R^\top + A_L\dot{B}^\top = \left(U_{r,2r}\Sigma_{r,2r}\right) B_R^\top + A_L\left(\Sigma_{1,r}V_{1,r}\right)^\top = \mathrm{truncSVD}\left(\nabla\mathcal{L}, 2r\right).
\end{aligned}$$

Note, that $\dot{A}^\top A_L = 0$. So the initialization $\Delta W_*^{(0)}$ ensures that $\mathrm{truncSVD}\left(\nabla\mathcal{L}, 2r\right)$ lies in the tangent space $\mathcal{T}_{\Delta W_*^{(0)}}\mathcal{M}_r$, which means that the first step of the RiemannLoRA Algorithm 6 will get the Riemannian gradient equal to $\mathrm{truncSVD}\left(\nabla\mathcal{L}, 2r\right)$.

To generalize the result, we should change the parametrization of the tangent space. Because of the fact that each of the tangent vectors ξ lies in $\mathcal{M}_{\leq 2r}$, we may use an unconstrained skeleton decomposition (without constraints for $\dot{A}$ and A_L):

$$\xi_* = \begin{bmatrix} A_*, \dot{A}_* \end{bmatrix} \begin{bmatrix} \dot{B}_*^\top \\ B_*^\top \end{bmatrix} = \begin{bmatrix} A_*, \dot{A}_* \end{bmatrix} I_{2r} \begin{bmatrix} \dot{B}_*^\top \\ B_*^\top \end{bmatrix} =$$

$$= \begin{bmatrix} A_*, \dot{A}_* \end{bmatrix} S S^{-1} \begin{bmatrix} \dot{B}_*^\top \\ B_*^\top \end{bmatrix}, \quad S \in \mathbb{R}^{2r \times 2r}, \det S \neq 0.$$

Representing S and its inverse as block matrices:

$$S = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix}, \quad S^{-1} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix},$$

one arrives to 15:

$$\xi_* = \begin{bmatrix} A_*, \dot{A}_* \end{bmatrix} \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} \dot{B}_* & B_* \end{bmatrix}^\top,$$

$$\Delta W_*^{(0)} = \begin{bmatrix} A'_*, \dot{A}'_* \end{bmatrix} \begin{bmatrix} S_{11} \\ S_{21} \end{bmatrix} \begin{bmatrix} C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} \dot{B}'_* & B'_* \end{bmatrix}^\top.$$

To derive the version that is utilized in practice, one may take S to be block-diagonal with $S_{11} = \alpha I_r, S_{22} = I_r, \alpha \in \mathbb{R}$. □

B RIEMANNION WITH LOI

The complete framework is summarized in Algorithm 5, which consists of an LOI initialization step followed by `max_iters` iterations of the Riemannion optimizer.

Specifically, the first 3 lines are dedicated to the computation of LOI via `BackPropRSVD` for given layer weights and oversampling parameter p, that increases the accuracy of the singular approximation. The 4-th step initializes Heavy-Ball momentum matrices. Therefore, the asymptotical complexity of this stage is determined by the complexity of Algorithm 3.

Algorithm 5 `Riemannion with LOI`

Require: Weights $W \in \mathbb{R}^{m \times n}$, rank $r \in \mathbb{N}$, step size η, momentum coefficient β, weight decay coefficient γ, oversampling parameter p, power-step parameter q.
Ensure: Tuning parameters $\Delta W^* \in \mathcal{M}_r$.
Function:

1: $A_L, \_, B^\top := \texttt{BackPropRSVD}(2r, p, q, \mathcal{L}, W)$. // $\mathcal{O}\left((m+n)r^2\right)$
2: $A_L, B^\top := A_L[:, :r],\ B[:, r:]^\top$.
3: $W' := W - A_L B^\top$. // $\mathcal{O}(mn)$
4: $A_{\mathrm{HB}}, B_{\mathrm{HB}} := 0, 0$.
5: **for** $i := 0, \ldots,$ `max_iters` **do**
6: $B_R := \texttt{qr}(B).\texttt{Q}$. // $\mathcal{O}\left(nr^2\right)$
7: $\dot{A} := \nabla_{Z_1} \mathcal{L}\left(W' + Z_1 B_R^\top + A_L Z_2^\top\right)|_{Z_1=0, Z_2=B}$. // $= \nabla_W \mathcal{L}(W) B_R$
8: $\dot{B} := \nabla_{Z_2} \mathcal{L}\left(W' + Z_1 B_R^\top + A_L Z_2^\top\right)|_{Z_1=0, Z_2=B}$. // $= \nabla_W \mathcal{L}(W)^\top A_L$
9: $\dot{A}_{\mathrm{prev}}, \dot{B}_{\mathrm{prev}} := \texttt{ProjectLR}\left((A_{\mathrm{M}}, B_{\mathrm{M}}), A_L, B_R\right)$. // $\mathcal{O}\left((m+n)r^2\right)$
10: $\dot{A}, \dot{B} := \beta \dot{A}_{\mathrm{prev}} + (I - A_L A_L^\top)\dot{A},\ \beta \dot{B}_{\mathrm{prev}} + \dot{B}$ // $\mathcal{O}\left((n+m)r^2\right)$
11: $\dot{A}, \dot{B} := \texttt{ProjectLR}\left(\texttt{OrthoLR}\left(A_L, B_R, \dot{A}, \dot{B}, r, \right), A_L, B_R\right)$ // $\mathcal{O}\left((m+n)r^2 + r^3\right)$
12: $U, \Sigma, V^\top := \texttt{RetractionLR}\left(\left[-\eta \dot{A}, A_L\right], \left[B_R, -\eta(\dot{B} + \gamma B_R)\right]\right)$ // $\mathcal{O}\left((m+n)r^2 + r^3\right)$
13: $A_{\mathrm{HB}}, B_{\mathrm{HB}} := [\dot{A}, A_L],\ [B_R, \dot{B}]$
14: $A_L, B := U, \Sigma V^\top$ // $\mathcal{O}\left(nr^2\right)$
15: **return** A_L, B.

C RIEMANN-SGD ALGORITHM

In Algorithm 6, we present the RiemannSGD version of the fine-tuning algorithm. The first steps are dedicated to the computation of the optimal initialization using `BackPropRSVD`(Algorithm 3). After initialization, the algorithm follows an Adam-like procedure adapted to the Riemannian setting: it maintains exponentially smoothed estimates of momentum and gradient norms ($4th$ step). The overall cycle encapsulates the computation of the Riemannian gradient at the current point(steps $7-8$), the update of the adaptive momentum terms via `ProjectLR`(Algorithm 2), and the retraction step that projects the updated tangent direction back onto the manifold(steps $15-17$). This approach has asymptotical complexity of $\mathcal{O}((m+n)r^2+r^3)$ and $(2(q+1)+\texttt{max\_iters})$ amount of backward calls.

Algorithm 6 `RiemannSGD with LOI`

Require: Weights $W \in \mathbb{R}^{m\times n}$, rank $r \in \mathbb{N}$, step size η, momentum coefficient β, oversampling parameter p, power-step parameter q, `simulate_Adam`, Adam momentum coefficient γ.
Ensure: Tuning parameters $\Delta W^* \in \mathcal{M}_r$.
Function:

$A_L, \_, B^\top := \texttt{BackPropRSVD}(2r, p, q, \mathcal{L}, W)$. // $\mathcal{O}\left((m+n)r^2\right)$
$A_L, B^\top := A_L[:, :r],\ B[:, r:]^\top$.
$W' := W - A_L B^\top$. // $\mathcal{O}(mn)$
$A_{\text{HB}}, B_{\text{HB}}, S_A, S_B := 0$.
for $i := 0, \ldots,$ `max_iters` **do**
  $B_R := \texttt{qr}(B).\texttt{Q}$. // $\mathcal{O}\left(nr^2\right)$
  $\dot{A} := \nabla_{Z_1}\mathcal{L}\left(W + Z_1 B_R^\top + A_L Z_2^\top\right)|_{Z_1=0, Z_2=B}$.
  $\dot{B} := \nabla_{Z_2}\mathcal{L}\left(W + Z_1 B_R^\top + A_L Z_2^\top\right)|_{Z_1=0, Z_2=B}$.
  $\dot{A}_{\text{prev}}, \dot{B}_{\text{prev}} := \texttt{ProjectLR}\left((A_{\text{HB}}, B_{\text{HB}}), A_L, B_R\right)$. // $\mathcal{O}\left((m+n)r^2\right)$
  $\dot{A}, \dot{B} := \beta\dot{A}_{\text{prev}} + (1-\beta)(I - A_L A_L^\top)\dot{A},\ \beta\dot{B}_{\text{prev}} + (1-\beta)\dot{B}$ // $\mathcal{O}\left(nr^2\right)$
  if `simulate_Adam` **then**
    $S_A, S_B := \gamma\|\dot{A}\|_F + (1-\gamma)S_A,\ \gamma\|\dot{B}\|_F + (1-\gamma)S_B$ // $\mathcal{O}\left((m+n)r\right)$
    $\dot{A}, \dot{B} := \dot{A}/S_A, \dot{B}/S_B$ // $\mathcal{O}\left((m+n)r\right)$
  $U, \Sigma, V^\top := \texttt{RetractionLR}\left(\left[\eta\dot{A}, A_L\right], \left[B_R, \eta\dot{B} + B\right]\right)$ // $\mathcal{O}\left((m+n)r^2 + r^3\right)$
  $A_{\text{HB}}, B_{\text{HB}} := [\dot{A}, A_L],\ [B_R, \dot{B}]$
  $A_L, B := U, \Sigma V^\top$ // $\mathcal{O}\left(nr^2\right)$
return A_L, B.

Table 2: The average accuracy (in %) among 8 tasks of fine-tuned Llama 3.2-1b using different SGD variants, tested on Commonsense Reasoning benchmark. The «R-» prefix stands for Riemannian, the postfix «-LOI» stands for locally optimal initialization. LoRA rank is set to 16. In all experiments excpet for «-LOI» the A factor in $AB^\top$ has orthonormal columns and B is zero.

Task Initialization	BoolQ	PIQA	SIQA	hella- swag	wino- grande	ARC-E	ARC-C	OBQA	All
Raw	40.1	55.4	50.3	25.8	50.0	61.9	41.8	42.8	46.0
LoRA	$64.0_{\pm 0.2}$	$75.3_{\pm 0.3}$	$69.6_{\pm 0.4}$	$82.2_{\pm 0.1}$	$53.0_{\pm 1.0}$	$75.4_{\pm 1.7}$	$56.5_{\pm 0.6}$	$67.1_{\pm 2.3}$	$67.9_{\pm 0.4}$
LoRA-LOI	$64.9_{\pm 0.4}$	$76.8_{\pm 0.3}$	$71.2_{\pm 2.2}$	$84.1_{\pm 0.1}$	$56.7_{\pm 0.6}$	$77.0_{\pm 4.2}$	$61.1_{\pm 3.6}$	$68.9_{\pm 3.3}$	$70.1_{\pm 1.6}$
RSLoRA	$64.5_{\pm 0.2}$	$77.2_{\pm 0.3}$	$68.1_{\pm 0.8}$	$86.3_{\pm 0.1}$	$58.2_{\pm 0.5}$	$76.5_{\pm 1.5}$	$58.6_{\pm 2.9}$	$70.0_{\pm 1.9}$	$70.0_{\pm 0.8}$
RiemannLoRA	$\mathbf{65.1}_{\pm 0.4}$	$77.7_{\pm 0.3}$	$69.1_{\pm 1.5}$	$85.8_{\pm 0.2}$	$58.5_{\pm 0.4}$	$74.4_{\pm 1.9}$	$56.9_{\pm 1.3}$	$69.3_{\pm 2.5}$	$69.6_{\pm 0.9}$
RiemannLoRA-LOI	$\mathbf{65.2}_{\pm 0.4}$	$\mathbf{79.4}_{\pm 0.4}$	$\mathbf{75.6}_{\pm 0.4}$	$\mathbf{87.3}_{\pm 0.1}$	$\mathbf{62.4}_{\pm 0.4}$	$\mathbf{79.9}_{\pm 0.8}$	$\mathbf{63.6}_{\pm 1.9}$	$\mathbf{73.8}_{\pm 0.5}$	$\mathbf{73.4}_{\pm 0.3}$

D DATASET

D.1 COMMONSENSE REASONING

Following the approach of Hu et al. (2023), we combine the training datasets from all 8 tasks to form the final training set and evaluate performance on each task's individual test dataset. We adopt

queries structure from Hu et al. (2023) to Llama 3.2 instruct template. The prompt structure is demonstrated in Table 3.

E HYPERPARAMETERS

For each optimization method, we carefully preselected the learning rate (step size). For the Riemannion method, we set the momentum to 0.9, and for the Muon method, we set the momentum to 0.95. In both cases, we additionally tuned the weight-decay hyperparameter (see Algorithm 4). The final choices of these hyperparameters are summarized in Table 4.

In the Commonsense reasoning benchmark, the hyperparameters are reported in Table 5 for the SGD-like methods and in Table 4 for the Adam-like methods. For the LOI method, we selected the following hyperparameters: the backprop RSVD oversampling parameter was set to $r_k = 16$, the backprop powerstep parameter was set to $q = 1$, and the multiplier α was set to $\frac{-0.01}{\sqrt{r}}$.

The non-tuned hyperparameters used for experiments on the Commonsense Reasoning dataset are presented in 6.

F SUBJECT-DRIVEN GENERATION

Training details We used Stable Diffusion-2-base model with a batch size of 4 for all experiments. For both methods, we set the betas to 0.9 and 0.999 and the weight decay to 0.1. In all variations of our approach, we set $q = 15$, p = rank and $\alpha = -\frac{1}{\sqrt{\text{rank}}}$. We used a learning rate of 2e-5 to train both our and the LoRA models, which were used to generate images shown in Figures 2 and 3.

Evaluation details We used the DreamBooth dataset, which contains 25 different prompts and 30 various concepts. Due to computational costs, we only used half of the proposed concepts to evaluate metrics: `can`, `candle`, `cat`, `cat2`, `colorful_sneaker`, `dog2`, `dog3`, `dog5`, `dog6`, `dog7`, `dog8`, `fancy_boot`, `grey_sloth_plushie`, `pink_sunglasses`, `vase`. To measure the similarity between the original concept and the generated images, we used Image Similarity (IS): we synthesized 30 images for the base prompt «a photo of a V*» and 10 images for each of 25 editing prompts (like «a V* on top of a dirt road»). We then measured the average pairwise cosine similarity with reference photos of the concept using the DINO model. To check the correspondence between the generated images and the textual prompts, we calculated Text Similarity (TS): we evaluated each concept with each of 25 prompts, synthesizing 10 images per prompt, and calculating the average pairwise cosine similarity with the prompts using the CLIP ViTB/32 model.

Additional results Figure 3 shows an additional visual comparison of our method and LoRA after 600 training steps. As can be seen, our method learns the concept much faster than the original LoRA, while preserving editing capabilities.

G ADDITIONAL EXPERIMENTAL RESULTS

G.1 ABLATION STUDY ON INITIALIZATION

The fine-tuning optimization was carried out by AdamW (Loshchilov & Hutter (2019)) First of the first, for every approach tested we preselected a suitable optimization step sizes. The list of all selected hyper-parameters for each method is specified in the Appendix E. Table 7 contains the accuracy of the trained model's responses on the test dataset. The notation «LoRA-A» in the table means the utilization of the vanilla LoRA with non-zero A, the notation «LoRA-B» means the same with non-zero B, the notations «Stiefel-A» and «Stiefel-B» indicate vanilla LoRA with orthonormal initialization (for matrix A and B respectively). The naming of «Stiefel-both» involves taking the factors A, B with orthonormal columns and with initialization like in (13).

Table 3: The structure of the queries for the Commonsense reasoning dataset

Task	Role	Fine-tuning Data Template
BoolQ	system	Please answer the following question with True or False. Follow the answer format, full answer not needed.
	user	Question: [QUESTION] Answer format: True/False
	assistant	The correct answer is [ANSWER]
PIQA	system	Please choose the correct solution to the question. Follow the answer format, full answer not needed.
	user	Question: [QUESTION] Solution1: [SOLUTION_1] Solution2: [SOLUTION_2] Answer format: Solution1/Solution2
	assistant	The correct answer is [ANSWER]
SIQA	system	Please choose the correct answer to the question based on the context provided. Follow the answer format, full answer not needed.
	user	Context: [CONTEXT] Question: [QUESTION] A: [ANSWER_A] B: [ANSWER_B] C: [ANSWER_C] Answer format: A/B/C
	assistant	The correct answer is [ANSWER]
hellaswag	system	Please choose the correct ending to complete the given sentence. Follow the answer format, full answer not needed.
	user	[ACTIVITY_lABEL]: [CONTEXT] Ending1: [ENDING_1] Ending2: [ENDING_2] Ending3: [ENDING_3] Ending4: [ENDING_4] Answer format: Ending1/Ending2/Ending3/Ending4
	assistant	The correct answer is [ANSWER]
winogrande	system	Please choose the correct answer to fill in the blank to complete the given sentence. Follow the answer format, full answer not needed.
	user	Sentence: [SENTENCE] Option1: [OPTION_1] Option2: [OPTION_2] Answer format: Option1/Option2
	assistant	The correct answer is [ANSWER]
ARC-e & ARC-c	system	Please choose the correct answer to the question. Follow the answer format, full answer not needed.
	user	Question: [QUESTION] Answer1: [ANSWER_1] Answer2: [ANSWER_2] Answer3: [ANSWER_3] Answer4: [ANSWER_4] Answer format: Answer1/Answer2/Answer3/Answer4
	assistant	The correct answer is [ANSWER]
OBQA	system	Please choose the correct answer to the question. Follow the answer format, full answer not needed.
	user	Question: [QUESTION] Answer1: [ANSWER_1] Answer2: [ANSWER_2] Answer3: [ANSWER_3] Answer4: [ANSWER_4] Answer format: Answer1/Answer2/Answer3/Answer4
	assistant	The correct answer is [ANSWER]

Table 4: The parameters for different Adam variants for fine-tuning on the Commonsense reasoning dataset

Optimizer	learning rate	weight decay
Adam	0.0002	0.01
DoRA	0.0003	0.01
Muon	0.0005	0.2
DoneRITE	0.0005	0.0001
RPrecAdamW	0.0005	0.5
Riemannion	0.0001	0.00316

Table 5: The parameters for different SGD variants (RiemannLoRA with `simulate_Adam` flag disabled) for fine-tuning on the Commonsense reasoning dataset

Optimizer	learning rate
LoRA	0.1
LoRA-LOI	0.06
RSLoRA	0.1
RiemannLoRA	0.1
RiemannLoRA-LOI	0.07

Table 6: Other non-tuned hyperparameter configurations for experiments on Commonsense reasoning dataset

Dataset Hyperparameters	Commonsense reasoning
Rank r	16
Dropout	0.05
LR Scheduler	Linear
Batch size	64
Epochs	2
Warmup ratio	0.1

Table 7: The average accuracy among 8 tasks of fine-tuned Llama 3.2-1b via Adam using different LoRA initialization variants, tested on Commonsense reasoning benchmark. «LOI» stands for locally optimal initialization. LoRA rank is set to 16.

Task Initialization	BoolQ	PIQA	SIQA	hella-swag	wino-grande	ARC-E	ARC-C	OBQA	All
Raw	40.1	55.4	50.3	25.8	50.0	61.9	41.8	42.8	46.0
LoRA-A	66.2	80.2	76.0	87.0	65.1	78.4	63.6	76.3	74.1
LoRA-B	65.9	77.9	74.2	82.9	61.2	73.9	60.6	72.2	71.1
Pissa	66.4	80.1	75.6	87.6	63.1	77.6	64.3	75.0	73.7
Stiefel-A	65.4	79.5	74.9	87.2	62.4	79.3	62.3	75.5	73.3
Stiefel-B	66.4	80.6	76.0	87.5	64.3	78.5	64.9	74.6	74.1
Stiefel-both	66.3	79.7	75.8	86.4	64.0	78.3	62.4	73.9	73.4
LOI	65.6	81.3	75.7	87.7	65.7	77.9	65.0	75.2	74.3
LoRA-GA	65.4	77.1	74.9	84.5	58.3	75.4	61.9	72.2	71.2

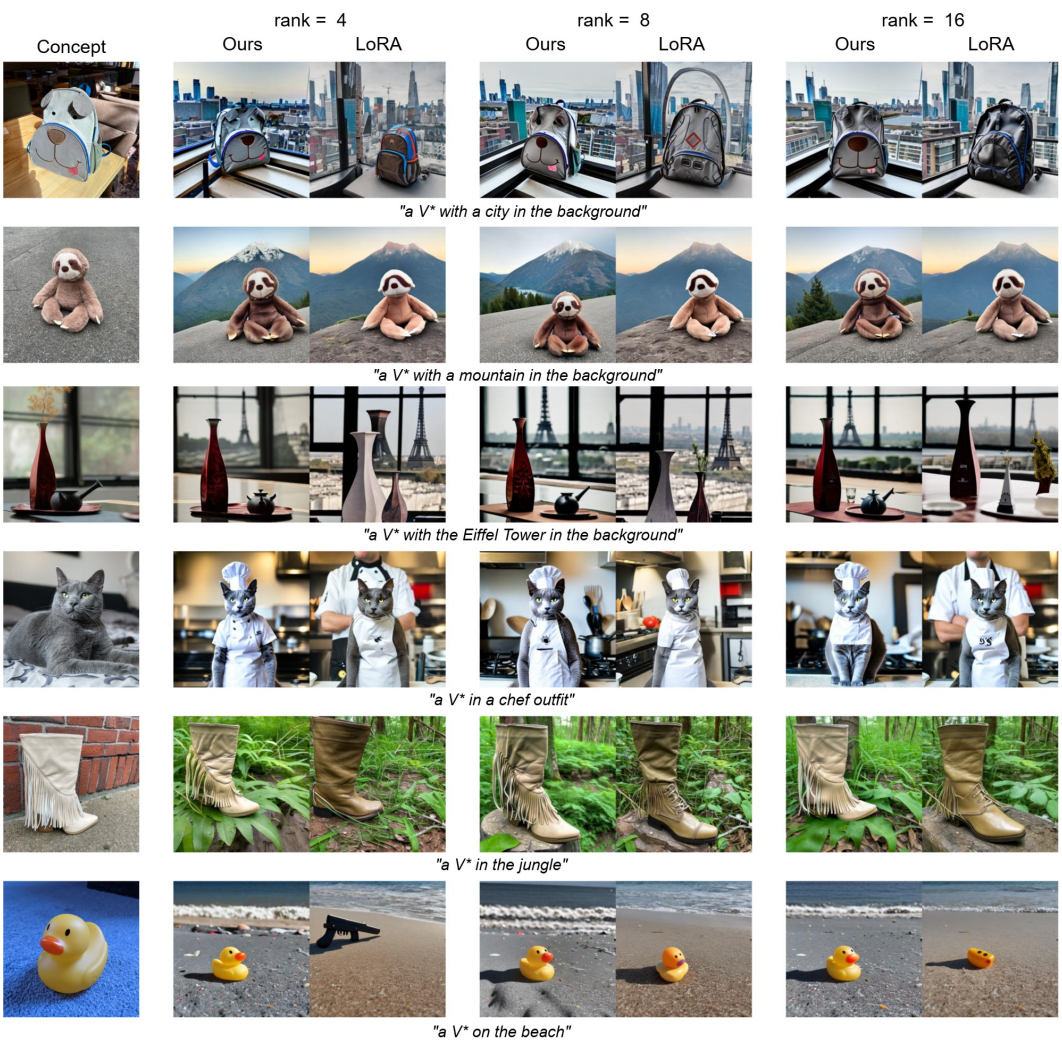

Figure 3: Additional visual comparison of our method and LoRA, checkpoint 600

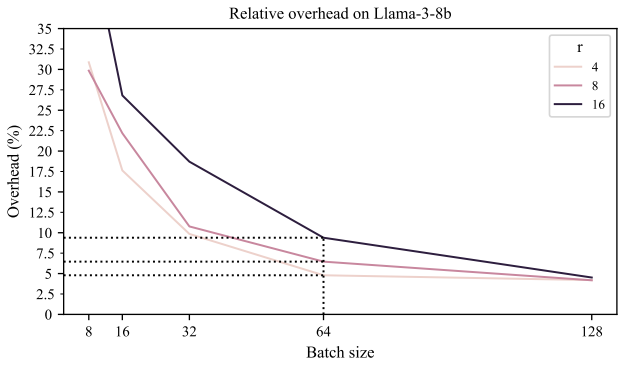

Figure 4: Relative Time Cost $\left(\text{calculated as } \left(T_{\text{Riemannion}} - T_{\text{Adam}}\right) / T_{\text{Adam}}\right)$ of Riemannion vs. Adam during Llama 3-8B fine-tuning, as a function of LoRA rank and batch size.

H COMPUTATIONAL COST

To measure execution time, we used a virtual server instantiated on a compute node equipped with an `Intel Xeon Gold 6240R` CPU and `Nvidia Tesla V100` GPU. The virtual machine was provisioned with 8 CPU cores, 16 GB of RAM and a single Tesla V100 GPU. Experiments were executed using the Hugging Face `transformers` library; timing corresponds to the execution time of the `trainer.train()` call. Measurements were conducted according to the following procedure. For each combination of method, LoRA adapter rank, and batch size, four timing runs were performed and the minimum execution time was selected. The measured quantity was the time spent on 16 optimizer steps. Time spent on initialization was excluded. For each combination of rank and batch size, execution times for each of the methods were measured sequentially. In the present work, rank refers to the rank of the LoRA adapter, and method denotes one of two optimization schemes: Adam (baseline) and Riemannion (proposed method).

Figure 4 shows the relative increase in per-step execution time of the proposed Riemannion method compared with Adam, computed as

I LLM USAGE

We used an LLM only for minor language polishing to improve readability and grammar; it was not involved in research ideation, methodology, analysis, or substantive writing, and all research ideas and arguments were developed entirely by the authors.