GEOLORA: GEOMETRIC INTEGRATION FOR PARAMETER EFFICIENT FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-Rank Adaptation (LoRA) has become a widely used method for parameter-efficient fine-tuning of large-scale, pre-trained neural networks. However, LoRA and its extensions face several challenges, including the need for rank adaptivity, robustness, and computational efficiency during the fine-tuning process. We introduce GeoLoRA, a novel approach that addresses these limitations by leveraging dynamical low-rank approximation theory. GeoLoRA requires only a single backpropagation pass over the small-rank adapters, significantly reducing computational cost as compared to similar dynamical low-rank training methods and making it faster than popular baselines such as AdaLoRA. This allows GeoLoRA to efficiently adapt the allocated parameter budget across the model, achieving smaller low-rank adapters compared to heuristic methods like AdaLoRA and LoRA, while maintaining critical convergence, descent, and error-bound theoretical guarantees. The resulting method is not only more efficient but also more robust to varying hyperparameter settings. We demonstrate the effectiveness of GeoLoRA on several state-of-the-art benchmarks, showing that it outperforms existing methods in both accuracy and computational efficiency.

1 INTRODUCTION

Large-scale pre-trained and fine-tuned models have significantly advanced the performance of deep learning models in assisting various natural language processing and computer vision tasks. However, their deployment often incurs substantial computational and memory costs due to the enormous number of trainable parameters. To address this, parameter-efficient fine-tuning (PEFT) methods have been developed, which modify a subset of model parameters while keeping the rest frozen. Among these, low-rank adaptation (LoRA) (Hu et al., 2021) has emerged as a prominent approach, allowing efficient fine-tuning by injecting low-rank updates into pre-trained model weights. Despite its efficiency, LoRA faces limitations in adaptively distributing the parameter budget across weight matrices, and its performance is sensitive to the choice of hyperparameters (Zhang et al., 2023).

Recent works, such as AdaLoRA (Zhang et al., 2023), DyLoRA (Valipour et al., 2023), and ReLoRA (Lialin et al., 2023), have attempted to improve LoRA by dynamically adjusting the rank of the low-rank adapters during training. While these methods enhance parameter efficiency, they are constructed as simultaneous descent methods and therefore do not guarantee convergence to optimal low-rank adapters. Methods that guarantee convergence to optimal adapters exist (Schotthöfer et al., 2022; Schotthöfer & Laiu, 2024; Zangrando et al., 2024). However, these require several gradient tapes per iteration and, therefore, have an intrinsically higher run time per training step.

In this paper, we introduce GeoLoRA (Geometric Low-Rank Adaptation), a novel dynamical low-rank training method for parameter-efficient fine-tuning. GeoLoRA leverages the dynamical low-rank approximation theory from matrix differential equations (Koch & Lubich, 2007b; Ceruti et al., 2022; 2023) and exploits the intrinsic low-rank geometry of the weight matrices to allocate the parameter budget across the model adaptively. This dynamic allocation is facilitated by a novel training strategy that updates the low-rank factors in parallel, contrasting with other recent methods based on dynamical low-rank approximation theory (Schotthöfer et al., 2022; Schotthöfer & Laiu, 2024; Zangrando et al., 2024), which require individual gradient tapes computed sequentially per each low-rank factor. Instead, GeoLoRA requires a single backprop pass over the small-rank adapters, limiting its computational cost and making it faster than popular baselines such as AdaLoRA (Zhang et al., 2023). Moreover, GeoLoRA maintains the exact orthonormality of the low-rank factors, avoiding the ill-conditioning issues associated with well-known high-curvature challenges arising in low-rank optimization (Schotthöfer et al., 2022).

Through extensive experiments on the GLUE benchmark, Vision Transformers, and Stable Diffusion, we show that GeoLoRA outperforms existing PEFT methods both in terms of accuracy and computational efficiency.

Along with the experimental evaluation, we provide a thorough convergence analysis, showing convergence to stationary points under standard assumptions, and a detailed error-bound analysis, demonstrating that GeoLoRA's low-rank adaptation remains close to its full-rank counterpart throughout the training process. This robustness is critical in ensuring that the fine-tuning process does not diverge, even under challenging conditions.

Overall, the main contributions of this work are as follows:

- We show that standard common training methods for low-rank adapters do not necessarily reach a local optimum. (Section 3)
- We propose GeoLoRA, a dynamical low-rank training method for low-rank adapters that leverages low-rank geometry and matrix differential equations to achieve adaptive parameter allocation. (Section 4)
- GeoLoRA only requires a single gradient tape and one small-size SVD per training step, making it competitive with existing baselines such as AdaLoRA.
- We provide a convergence analysis and error bound guarantees for GeoLoRA, ensuring robust training behaviour and convergence to a stationary point. (Section 4.2)
- Extensive experimental results demonstrate the superior performance of GeoLoRA over existing methods, with improved accuracy and training speed. (Section 5)

2 RELATED WORK

The growing size of neural networks has led to significant computational and memory challenges during both training and deployment. Several strategies have been proposed to mitigate these issues, including sparsification (Guo et al., 2016; Molchanov et al., 2017; He et al., 2017) and quantization (Wu et al., 2016; Courbariaux et al., 2016). Among these, layer factorization has gained traction as an effective approach to reducing memory requirements. Layer factorization techniques have been applied successfully in both pre-training (Wang et al., 2021; Khodak et al., 2021; Schotthöfer et al., 2022; Schotthöfer & Laiu, 2024; Zangrando et al., 2024; Zhao et al., 2024) and fine-tuning scenarios (Hu et al., 2021; Valipour et al., 2023; Zhang et al., 2023; Hayou et al., 2024; Zhao et al., 2024; Lialin et al., 2023), demonstrating their versatility across various tasks.

Low-rank adapters such as LoRA (Hu et al., 2021) have become a standard approach for PEFT by applying low-rank corrections to pre-trained models. LoRA introduces a low-rank decomposition to the weight matrices of the model, significantly reducing the number of trainable parameters while preserving performance. Despite its efficiency, LoRA's effectiveness heavily relies on the selection of hyperparameters such as learning rates and parameter budgets (Zhang et al., 2023; Hayou et al., 2024). These limitations have spurred the development of rank-adaptive methods. AdaLoRA (Zhang et al., 2023) is a popular extension of LoRA, which dynamically adjusts the rank of the low-rank adapters during training. By incorporating an orthogonality regularizer and SVD-like adaptation, AdaLoRA aims to address the challenges of rank selection and adaptation. It outperforms static low-rank methods by automatically allocating parameter budgets based on the importance of each matrix component. DyLoRA (Valipour et al., 2023) provides an alternative approach that hierarchically adjusts the rank during training, demonstrating that higher-rank adapters can lead to better performance than very low-rank ones. DoRA (Mao et al., 2024) proposes to sample a set of rank-1 updates for each LoRA layer and to combine them into a rank-r update. Optimal rank-1 components are chosen during fine-tuning using an importance score based on the norm of the LoRA layer.

Beyond fine-tuning, low-rank methods have been successfully applied during the training and pre-training phases of neural networks. Techniques such as Pufferfish (Wang et al., 2021), intrinsic dimension reduction (Aghajanyan et al., 2020), and DLRT (Schotthöfer et al., 2022) suggest that large deep learning models have an inherently low intrinsic dimensionality, making them amenable to low-rank approximations. These methods propose reducing the number of parameters during training, potentially improving both efficiency and generalization. Recent works in dynamical low-rank training have explored the use of geometric properties of the low-rank parameter space to improve training stability and convergence. For example, the geometry-aware training approach for tensor layers in Tucker format (Zangrando et al., 2024) dynamically adapts the rank of the factorized layers, ensuring robust convergence even when the initial rank estimation is inaccurate. This method leverages the Riemannian geometry of the parameter space to avoid the ill-conditioning commonly encountered in low-rank training. ReLoRA (Lialin et al., 2023) introduces a parameter-efficient training method by using multiple low-rank updates to effectively train high-rank networks. This method allows training larger models with significant memory savings and training speed improvements compared to conventional methods. GaLore (Zhao et al., 2024) introduces a memory-efficient training strategy by projecting gradients onto a low-rank subspace. This approach achieves significant memory savings while maintaining performance.

3 LOW-RANK OPTIMIZATION: WHAT CAN GO WRONG

This section aims to discuss the nature of the critical points and optimization trajectories obtained when using gradient-based strategies for low-rank parameters, and why a straightforward application of gradient-based steps to factorized adapters may lead to suboptimal results.

Consider a neural network layer of the form

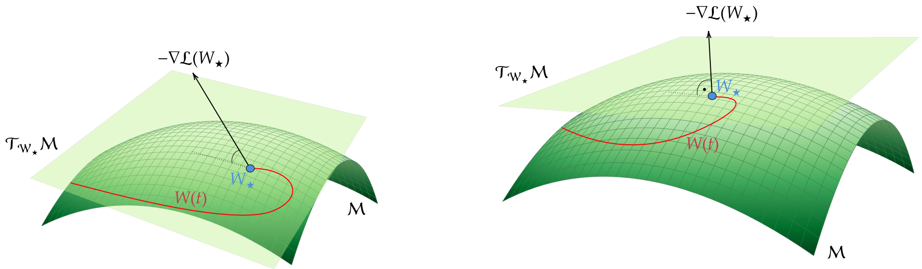
$$\mathbf{z} = \sigma(W_{\mathrm{pt}}\mathbf{x} + USV^\top \mathbf{x}), \tag{1}$$

where σ is an arbitrary activation function, $W_{\mathrm{pt}} \in \mathbb{R}^{n\times n}$ are the frozen pre-trained weights, and $U, V \in \mathbb{R}^{n\times r}$, $S \in \mathbb{R}^{r\times r}$ are the rank-r adapter weights, with input $\mathbf{x}$. For simplicity, we omit the bias term. Low-rank adapters of the form $W = USV^\top \in \mathbb{R}^{n\times n}$ have gained popularity in recent approaches such as (Zhang et al., 2023), although our discussion extends to other equivalent formulations like $W = AB$ (Hu et al., 2021). The objective of the training process is to minimize a loss function $\mathcal{L}(W)$ to find an optimal adapter weight $W_\star$. For full-rank matrices ($r = n$), optimality requires that $\nabla_W \mathcal{L}(W_\star) = 0$. However, when $r < n$, this condition is generally unattainable due to the reduced parameter space. In this scenario, we seek a matrix $W_\star$ that is locally optimal within the low-rank parameter space, meaning no further reduction in the loss function $\mathcal{L}$ is possible in the neighborhood of $W_\star$. A necessary condition for local optimality can be expressed as $P(W_\star)\nabla\mathcal{L}(W_\star) = 0$, see e.g., (Sato, 2021, Theorem 3.4). For orthonormal U and V, the projection operator $P(USV^\top)Z := UU^\top Z(I - VV^\top) + ZVV^\top$ represents the *orthonormal* projection of Z onto the tangent space at $USV^\top$. If $W_\star$ is not a saddle point, then this condition ensures that no search direction within the tangent space of $W_\star$ can further decrease the loss. See also Appendix J. Note that this only guarantees local optimality, a limitation shared by all gradient-based optimizers.

Current training methods for low-rank adapters aim to optimize the low-rank factors with a single backpropagation pass to compute all the required gradients simultaneously. This boils down to integrating the following gradient flow equations for each individual factor

$$\begin{aligned}
\dot{U} &= -\nabla_U \mathcal{L} = -(\nabla_W \mathcal{L})VS\,, \\
\dot{V} &= -\nabla_V \mathcal{L} = -(\nabla_W \mathcal{L})^\top US^\top\,, \\
\dot{S} &= -\nabla_S \mathcal{L} = -U^\top \nabla_W \mathcal{L} V\,,
\end{aligned} \tag{2}$$

where we use the chain rule and the decomposition $W = USV^\top$ to derive the expressions for $\nabla_{U,S,V}\mathcal{L}$. Here, we have omitted the dependence on the time variable t, i.e., $U, S, V = U(t), S(t), V(t)$ for improved readability, and we use dots to denote time derivatives. An explicit time discretization with a time step size equal to the learning rate λ leads to the simultaneous gradient descent updates commonly employed in conventional training methods for LoRA. At first glance, this procedure appears effective, as a single update step will decrease the loss *if we freeze all but one of the low-rank factors*. However, in practice, LoRA training modifies all low-rank factors *simultaneously*, raising the question of how this affects the overall optimization trajectory.


$-\nabla\mathcal{L}(W_\star)$
$\mathcal{T}_{W_\star}\mathcal{M}$
$W_\star$
$W(t)$
$\mathcal{M}$
$-\nabla\mathcal{L}(W_\star)$
$\mathcal{T}_{W_\star}\mathcal{M}$
$W_\star$
$W(t)$
$\mathcal{M}$

(a) Gradient flow of simultaneous gradient descent.

(b) Riemannian gradient flow.

Figure 1: Illustration of simultaneous vs. Riemannian gradient flow. The projector of the simultaneous gradient flow converges to a point $W_\star$ such that $\widehat{P}(W_\star)\nabla\mathcal{L} = 0$. Since $\widehat{P}$ is not an orthogonal projection, the gradient is not orthogonal to the tangent plane, i.e., $W_\star$ is suboptimal. For Riemannian gradient flows, the adapter converges to a point $W_\star$ such that $P(W_\star)\nabla\mathcal{L} = 0$. Since P is the orthogonal projection on the tangent space, $W_\star$ is a local optimum, i.e., no directions exist in the tangent space $\mathcal{T}_{W_\star}\mathcal{M}$, which further decrease the loss. Here, $\mathcal{M}$ denotes the space of low-rank adapters, and $\mathcal{T}_{W_\star}\mathcal{M}$ represents the tangent space at the optimal adapter weight $W_\star$.

To address this, consider the evolution equation for $W = USV^\top$, derived directly using the chain rule and eq. (2)

$$\begin{aligned}\dot{W} &= \dot{U}SV^\top + U\dot{S}V^\top + US\dot{V}^\top \\ &\overset{(2)}{=} -\nabla_W\mathcal{L}VS^2V^\top + UU^\top\nabla_W\mathcal{L}VV^\top - U(S^\top)^2U^\top\nabla_W\mathcal{L} =: -\widehat{P}(W)\nabla_W\mathcal{L}\,.\end{aligned} \tag{3}$$

The operator $\widehat{P}(USV^\top)Z := ZVS^2V^\top - UU^\top ZVV^\top + U(S^\top)^2U^\top Z$ again represents a linear mapping onto the tangent space at $W = USV^\top$. Note that this projection depends on the individual low-rank factors U, S, and V, but we use the notation $\widehat{P}(W)$ for brevity. Simultaneous descent methods approximate the gradient flow of eq. (3), which ideally converges to a solution $W_\star$ such that $\widehat{P}(W_\star)\nabla\mathcal{L}(W_\star) = 0$. However, $\widehat{P}$ is orthogonal only when U and V are orthonormal and $S = I$, where I denotes the identity matrix. If these conditions are not met, the resulting optimization process may not find an optimal weight within the low-rank parameter space. This is because $\widehat{P}(W_\star)\nabla\mathcal{L}(W_\star) = 0$ does not imply $P(W_\star)\nabla\mathcal{L}(W_\star) = 0$, thus there could still be some decrease direction along the tangent space as depicted in Figure 1.

To construct methods that converge to an optimal low-rank solution, an alternative approach is to evolve the adapter W along the projected gradient flow $\dot{W}(t) = -P(W(t))\nabla\mathcal{L}(W(t))$. In this case, the corresponding evolution equations for the low-rank factors take the form

$$\begin{aligned}\dot{U} &= -(I - UU^\top)\nabla_W\mathcal{L}VS^{-1}\,, \\ \dot{V} &= -(I - VV^\top)\nabla_W\mathcal{L}^\top US^{-\top}\,, \\ \dot{S} &= -U^\top\nabla_W\mathcal{L}V\,,\end{aligned} \tag{4}$$

assuming that U and V are orthonormal (Koch & Lubich, 2007b).

While the evolution defined in eq. (4) guarantees convergence to an optimal low-rank adapter, the presence of the S^{-1} term on the right-hand side introduces stiffness in the gradient flow. This stiffness can significantly slow down convergence, especially when the singular values in S vary greatly in magnitude. Robust solutions to address the stiffness problem and ensure convergence without being hindered by the S^{-1} term have been proposed Schotthöfer et al. (2022); Zangrando et al. (2024); Schotthöfer & Laiu (2024). However, these methods require multiple gradient tape evaluations per training update, which makes them computationally more expensive than traditional LoRA training techniques with simultaneous updates.

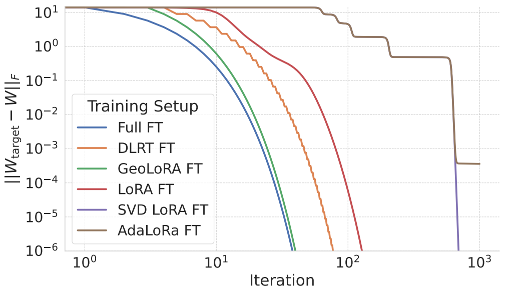
To overcome these limitations, we propose GeoLoRA, a novel training method for low-rank adapters that only requires a single gradient tape evaluation per update while ensuring convergence to an optimal low-rank solution, following the projected gradient flow in eq. (4). This approach retains the computational efficiency of conventional LoRA methods while achieving comparable or even superior performance. By eliminating the need for multiple gradient tape evaluations, GeoLoRA offers a practical and scalable solution for training low-rank adapters effectively.

Before presenting the proposed training method, we illustrate different behaviours of different low-rank adaptation strategies using a toy example. Consider the problem of matching a rank-r target matrix $W_{\text{target}} \in \mathbb{R}^{n\times n}$ with a low-rank adapter W, formulated as:

$$\min_{W} \frac{1}{2} \left\| W_{\text{target}} - W \right\|_F^2 . \tag{5}$$

We compare the convergence behavior of six different training methods for $n = 5000$, $r = 5$, and a learning rate of $\lambda = 0.1$:

1. Full fine-tuning (FT) (blue),
2. DLRT from (Schotthöfer et al., 2022) (orange),
3. The proposed GeoLoRA method (green),
4. Fixed rank LoRA from Hu et al. (2021) (red),
5. AdaLoRA from (Zhang et al., 2023) (brown),
6. Fixed rank AdaLoRA (purple).



In this experiment, the fixed rank approaches (4, 6) use a rank of 50. All adapters W are initialized to zero, with $S_0 = 0$ for the SVD-based methods (2, 3, 5, 6) and $B = 0$ for LoRA-based methods (4).

The results show that the proposed GeoLoRA method (3) converges as quickly as full fine-tuning. In contrast, method (2) (DLRT) takes approximately twice as long due to the sequential updates of the basis and coefficient matrices[1]. LoRA-type methods (4, 5, 6) exhibit slower convergence due to the suboptimality of the underlying gradient flow defined in eq. (2). AdaLoRA (5) solves the same gradient flow as method (6) but plateaus at a loss of 5×10^{-4}, corresponding to the regularization parameter for the terms $\|U^\top U - I\|_F^2 + \|V^\top V - I\|_F^2$ that enforce the orthonormality of U and V. We had to fix the minimum rank to 5 for AdaLoRA to prevent stalling of the optimization due to rank underestimation, an issue not observed in methods (2) and (3), where rank augmentation avoided this problem.

4 THE PROPOSED METHOD

In this section, we introduce GeoLoRA (Geometric Low-Rank Adaptation) a novel low-rank fine-tuning method that integrates **rank adaptivity**, **low-rank optimality**, and **memory and computational efficiency**. Our method builds upon the parallel geometric low-rank integrator originally designed for model order reduction in high-dimensional PDEs (Ceruti et al., 2023), and it is equipped with loss descent, approximation bounds, and convergence guarantees. Notably, it improves upon existing dynamical low-rank methods, e.g. (Schotthöfer & Laiu, 2024; Zangrando et al., 2024; Schotthöfer et al., 2022) by updating basis and coefficients *in parallel* opposed to a *sequential* basis update and coefficient step. Moreover, only a single backward pass per iteration step is required through a novel evaluation strategy of robust gradients, thus doubling the wall-time performance. GeoLoRA is, therefore, the first low-rank training method solving the optimal gradient flow eq. (4) with training times per iteration comparable to standard simultaneous descent approaches to low-rank adaptation such as LoRA and AdaLoRA (Hu et al., 2021; Zhang et al., 2023). In particular, it improves upon these methods by providing robustness and convergence guarantees and demonstrating an overall improved performance and robustness to hyperparameters in numerical examples.

Starting from an initial factorization U_0, V_0, S_0 with initial rank r_0, where S_0 is diagonal and full-rank, GeoLoRA performs the following steps (also summarized in Algorithm 1):

---

[1] The loss plateaus appear since the loss value remains constant during a basis update and only decreases during a coefficient update. This accounts for the fact that two gradient tapes need to be computed.

Algorithm 1: Single iteration of GeoLoRA.
The functions `optimizer_step`, `basis_augmentation`, and `truncation` are detailed in Algorithm 2 in the appendix.

```
Input : Initial orthonormal bases $U, V \in \mathbb{R}^{n \times r}$ and diagonal $S \in \mathbb{R}^{r \times r}$;
        $\tau$: singular value threshold for rank truncation;
        $\lambda$: learning rate.
Evaluate $\mathcal{L}(USV^\top)$                                             /* Forward evaluate */
$G_U \leftarrow \nabla_U \mathcal{L}(USV^\top)$; $G_S \leftarrow \nabla_S \mathcal{L}(USV^\top)$; $G_V \leftarrow \nabla_V \mathcal{L}(USV^\top)$   /* Backprop */
{ $S^{\text{new}} \leftarrow$ optimizer_step($S, G_S, \lambda$)
{ $K^{\text{new}} \leftarrow$ optimizer_step($US, G_U S^{-\top}, \lambda$)                 /* in parallel */
{ $L^{\text{new}} \leftarrow$ optimizer_step($VS^\top, G_V S^{-1}, \lambda$)
{ $\widetilde{U} \leftarrow$ basis_augmentation($U, K^{\text{new}}$)
{ $\widetilde{V} \leftarrow$ basis_augmentation($V, L^{\text{new}}$)                     /* in parallel */
$\widehat{S} \leftarrow \begin{bmatrix} S^{\text{new}} & L^{\text{new},\top}\widetilde{V} \\ \widetilde{U}^\top K^{\text{new}} & 0 \end{bmatrix} \in \mathbb{R}^{2r \times 2r}$   /* Assemble new coefficient matrix */
$U, S, V, S^{-1} \leftarrow$ truncation($\widehat{S}, [U \mid \widetilde{U}], [V \mid \widetilde{V}]$)
```

1. **Perform a (stochastic) gradient step** to compute the new variables $S^{\text{new}} \in \mathbb{R}^{r_0 \times r_0}$, $L^{\text{new}} \in \mathbb{R}^{n \times r_0}$, and $K^{\text{new}} \in \mathbb{R}^{n \times r_0}$, as follows:

$$\begin{aligned} S^{\text{new}} &= S_0 - \lambda \nabla_S \mathcal{L}(U_0 S_0 V_0^\top) \\ K^{\text{new}} &= U_0 S_0 - \lambda \nabla_U \mathcal{L}(U_0 S_0 V_0^\top) S_0^{-\top} \\ L^{\text{new}} &= V_0 S_0^\top - \lambda \nabla_V \mathcal{L}(U_0 S_0 V_0^\top) S_0^{-1}. \end{aligned} \tag{6}$$

We will see in Theorem 3 that using these variables mitigates the stiffness of the system in eq. (4) while approximating the optimal gradient flow. Note that the right-hand side gradients $\nabla_U \mathcal{L}$, $\nabla_V \mathcal{L}$, and $\nabla_S \mathcal{L}$ can be evaluated with only one backward pass through the network using standard algorithmic differentiation techniques, halving the computational cost of existing geometric methods such as (Schotthöfer et al., 2022; Zangrando et al., 2024). Evaluation of the inverse S_0^{-1} induces no computational overhead since S_0 is diagonal at the start of each iteration.

2. **Augment the current bases** U_0, V_0 to twice their rank using the gradient dynamics of the loss, which is encoded in K^{new} and L^{new}, i.e.

$$\widehat{U} = [U_0, \widetilde{U}] = \text{ortho}([U_0, K^{\text{new}}]) \in \mathbb{R}^{n \times 2r_0} \quad \text{and} \quad \widehat{V} = [V_0, \widetilde{V}] = \text{ortho}([V_0, L^{\text{new}}]) \in \mathbb{R}^{n \times 2r_0}. \tag{7}$$

Here "ortho" denotes a column orthonormalization procedure such as the QR-algorithm. This augmentation step provides the low-rank adapter with a larger search space to increase the rank of its adaptation if the initial rank-guess r_0 was insufficient to fully capture the problem. Doubling the rank implies that in $\log(n)$ training iterations any rank can be captured by a rank one initialization, eliminating the need for tuning r as a hyperparameter, see Figure 2.

3. **Assemble the augmented coefficient matrix**

$$\widehat{S} \leftarrow \begin{bmatrix} S^{\text{new}} & L^{\text{new},\top}\widetilde{V} \\ \widetilde{U}^\top K^{\text{new}} & 0 \end{bmatrix} \in \mathbb{R}^{2r_0 \times 2r_0} \tag{8}$$

where we obtain the block entries S^{new}, L^{new}, and K^{new} from eq. (6).

4. **Truncate redundant singular values** s_i of $\widehat{S}$ and the corresponding singular vectors, i.e. basis functions of $\widehat{U}, \widehat{V}$, using the criterion

$$\sum_{i=r_1+1}^{2r} s_i^2 < \vartheta, \tag{9}$$

where r_1 is the new rank of the factorization and ϑ is a tresholding hyperparameter. The singular values s_i are obtained via the SVD of $\widehat{S} = P\Sigma Q^\top \in \mathbb{R}^{2r_0 \times 2r_0}$. Then we determine the new factorization as

$S_1 = \mathrm{diag}(s_1, \dots, s_{r1}) \in \mathbb{R}^{r_1 \times r_1}$, $U_1 = \widehat{U} P_{(1,\dots,r_1)} \in \mathbb{R}^{n \times r_1}$ and $V_1 = \widehat{V} Q_{(1,\dots,r_1)} \in \mathbb{R}^{n \times r_1}$. The truncation threshold ϑ is chosen relative to the nuclear norm of the specific layer's current singular values, i.e. $\vartheta = \tau \|\widehat{S}\|_F^2$. Other norms, such as the 1-norm of the singular values s_i, are possible as well. Thus, the truncation threshold determines how aggressively to prune each layer individually. Analogously, the following global threshold similar to the one used in e.g. (Zhang et al., 2023; Ghadiri et al., 2023; Idelbayev & Carreira-Perpinan, 2020)

$$\sum_{\ell=1}^{L} \sum_{i=\ell+1}^{2r_\ell} s_{i,\ell}^2 < \frac{\tau}{1-\tau} \sum_{\ell=1}^{L} \sum_{i=1}^{r_{1,\ell}} s_{i,\ell}^2, \tag{10}$$

can be considered by summing the singular values across all the layers $\ell = 1, \dots, L$. To directly control the parameter budget, order $s_{i,\ell}^2$ by descending by magnitude and selecting the largest ones first until either eq. (10) is violated or the budget is depleted.

4.1 PARAMETER INITIALIZATION

LoRA-type adapters (Hu et al., 2021) initilize the low rank matrices B, A with zero initialization of B, and Gaussian initialization of A. This ensures that the fine-tuning indeed starts at the pretrained state of the network, i.e., $\sigma(W_{\mathrm{pt}}\mathbf{x} + \frac{\alpha}{r} A_0 B_0^\top \mathbf{x}) = \sigma(W_{\mathrm{pt}}\mathbf{x})$. For consistency with this initialization, the bases U_0 and V_0 can be initialized as random but orthonormal, whereas the coefficient matrix S_0 has zero-initialization. In this first solve of eq. (6), we set S_0^{-1} as the identity matrix. As a result, the first solve of eq. (6) is inconsistent with the optimal dynamics of eq. (4). However, all following iterations evolve the low-rank trajectory according to the optimal gradient flow. Since in the first iterations of a LoRA fine-tuning, the adapter is typically close to the original solution but far from the fine-tuning optimum, this inconsistency is irrelevant to the overall convergence behavior of the method. Alternatively, the required gradients can be computed with three individual gradient tapes in the first iteration, which does not require the inversion of S_0.

The proposed method can readily be used for **dynamic low-rank compression** (Schotthöfer et al., 2022; Zangrando et al., 2024) of pre-trained networks, where we consider a layer $\mathbf{z} = \sigma(W\mathbf{x})$, and approximate $W \approx U_0 S_0 V_0^\top$. Here, the initial parameters U_0, S_0, V_0 are obtained by a truncated singular value composition of W. Finally, for **low-rank pre-training** of an untrained network with given architecture, i.e. predetermined layer dimensions n, but unknown rank r, the factors U_0, V_0 are initialized randomly, but orthonormal and S_0 is initialized randomly, but diagonal for easy initialization of S_0^{-1}.

4.2 ANALYSIS

In the following, we analyze Algorithm 1 under the general assumption that $\mathcal{L}$ is L-smooth with constant L and bounded with constant B.

For brevity of exposition we denote $W_t^r = U_t S_t V_t^\top$ as the low-rank factorization at iteration t evaluated with Algorithm 1, whereas W_t denotes the full-rank solution obtained by "full fine-tuning" with stochastic gradient descent. Further, we denote by $f(W_t^r, \xi_t)$ the stochastic gradient of the network loss $\mathcal{L}$ w.r.t the low-rank weight W_t^r at iteration t, obtained by batch-gradient descent. The i.i.d random variable ξ_t models the randomness in the training data batch at iteration t. Lastly, recall that $P(W_t^r)Z$ denotes the orthogonal projection of the matrix Z onto the tangent plane of the manifold of rank-r matrices at the point W_t^r.

Algorithm 1 is an optimizer on low-rank manifolds: Theorem 1 shows, that the proposed scheme with stochastic gradients indeed decreases the training loss in each iteration, while optimizing on a manifold, and Theorem 2 yields stochastic convergence to a locally optimal stationary point.

Theorem 1 (Stochastic descent estimate)**.** *Algorithm 1 with stochastic (mini-batch) gradients fulfills*

$$\mathbb{E}_{\xi_{t+1}}[\mathcal{L}(W_{t+1}^r)] \leq \mathcal{L}(W_t^r) - \lambda \left(1 - \frac{L\lambda^2}{2}\right) \mathbb{E}_{\xi_1}[\|P(W_t^r) f(W_t^r, \xi_t)\|^2] + L\mathbb{E}_{\xi_1}[\|W_{t+1}^r - \widehat{W}_t^r\|]\,. \tag{11}$$

where W_t^r, $\widehat{W}_t^r$, W_{t+1}^r are the low-rank weight matrices at the start of iteration $t+1$, before, and after the truncation step, respectively.

The proof is provided in Appendix D. The above theorem yields a loss descent guarantee up to the two last terms on the right-hand side. The first term of the right hand side induces the step size criterion $\lambda \leq \frac{2}{L}$, which resembles the step size criterion of full gradient descent, where the two right hand side terms read $-\lambda(1-\frac{L\lambda}{2})\|f(W_t)\|^2$. This shows that the low-rank optimizer allows similar learning rates as a full fine-tuning setup, eliminating the need for the $\frac{\alpha}{r}$ scaling parameter of LoRA. The last term models the error introduced by the truncation step and is bounded by the user-determined cutoff threshold ϑ, as $\mathbb{E}_{\xi_1}[\|W_{t+1}^r - \widehat{W}_t^r\|] \approx \vartheta$. As the solution stabilizes in rank, the error term vanishes, and we obtain the following main convergence result:

Theorem 2 (Convergence). *Let $\mathcal{L} \geq 0$ and $W_1^r, \ldots, W_T^r$ be the solutions generated by Algorithm 1 over T steps. Let the learning rate sequence $\{\lambda_t\}$ satisfy the Robbins-Monro conditions*

$$\textstyle\sum_t \lambda_t = +\infty \qquad \sum_t \lambda_t^2 < +\infty\,,$$

and each step λ_t the step size restriction $\lambda_t \leq \frac{2}{L}$. Further assume $\sum_{t=1}^{T-1}\mathbb{E}[\|W_{t+1}^r - \widehat{W}_t^r\|] \leq D < \infty$, i.e. after some time, the solution W_t^r is contained in a manifold of rank r. Then we have

$$\liminf_{T\to\infty} \mathbb{E}[\|P(W_t^r)f(W_t^r)\|^2] = 0\,,$$

where the expected value is taken over all ξ_t.

The proof is provided in Appendix E. Additionally, the solution trajectory of Algorithm 1 is close to the (full-rank) trajectory of the dynamical system

$$\dot{W}(t) = -\nabla_W \mathcal{L}(W(t)), \tag{12}$$

i.e., the gradient flow of full training or fine-tuning:

Theorem 3 (Error-bound). *For an integer k, let $t = k\lambda$. Let $W(t)$ be the solution of eq. (12), and let W_t^r be the factorized low-rank solution after k steps with Algorithm 1. Assume that for any Z in a neighborhood of $W(t)$, we have $\|(I - P(Z))\nabla\mathcal{L}(Z)\| < \varepsilon$, i.e., the gradient flow is close to $T_Z\mathcal{M}_r$. Then,*

$$\|W(t) - W_t^r\| \leq c_1\varepsilon + c_2\lambda + c_3\vartheta/\lambda\,. \tag{13}$$

Moreover, let $W_{RF}(t)$ denote the solution of the Riemannian flow of eq. (4). Then,

$$\|W_{RF}(t) - W_t^r\| \leq c_4\varepsilon + c_2\lambda + c_3\vartheta/\lambda \tag{14}$$

where the constants c_1, c_2, c_3, c_4 depend only on L and B.

The proof is provided in Appendix G. We refer to Appendix F for an interpretation of Algorithm 1 as an integrator of the gradient flow of eq. (12).

Finally, we point out that the single-layer case discussed so far is not restrictive, and all the theoretical results above can be directly transferred to the multilayer setting by means of the following proposition:

Proposition 1 (Global structure preservation). *The application of Algorithm 1 for multiple LoRA layers corresponds to the numerical integration of an augmented single matrix system on the adjacency matrix of the computational graph*

$$\dot{\mathcal{W}} = -P(\mathcal{W})\Pi\nabla\mathcal{L}(\mathcal{W})$$

Where Π is a linear projection that depends only on the structure of the neural network architecture. Moreover, the application of Algorithm 1 to this system, leads to the global truncation strategy proposed in Section 4.

The proof of Proposition 1 can be found in Appendix I together with the relative derivation of the global truncation strategy.

5 NUMERICAL RESULTS

DeBERTa for GLUE. We evaluate the performance of GeoLoRA by fine-tuning the 183 million parameter transformer DeBERTaV3-base (He et al., 2023) on the GLUE Benchmark (Wang et al., 2019) and compare the results in Table 2. For details on the methods, implementation, hyperparameter choices, and benchmark setup, please refer to Appendix B.1. In most cases, GeoLoRA outperforms other methods on the benchmark, achieving better metrics with significantly fewer trainable parameters. This reduction in trainable parameters allows GeoLoRA to process substantially more samples during training and evaluation compared to AdaLoRA.

Table 1: Method comparison for low-rank finetuning. We compare the computational cost of a single training step for an $n \times n$ layer matrix of rank r. In the table, "local optimality" refers to the property $P(W_\star)\nabla\mathcal{L}(W_\star) = 0$ for the computed adapter $W_\star$, as discussed in Section 3.

Method	compute (per iteration)	memory (per iteration)	# gradient evals.	rank adaptive	local optimality
Full FT	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	1	-	✓
GeoLoRA	$\mathcal{O}(2nr + (2n+1)r^2 + r^3)$	$\mathcal{O}(4nr + 3r^2)$	1	✓	✓
AdaLoRA (Zhang et al., 2023)	$\mathcal{O}(2nr + (2n+1)r^2 + r^3)$	$\mathcal{O}(2nr + 3r^2)$	1	✓	✗
DLRT (Schotthöfer et al., 2022)	$\mathcal{O}(6nr + (2n+5)r^2 + 9r^3)$	$\mathcal{O}(4nr + 3r^2)$	3	✓	✓
LoRA (Hu et al., 2021)	$\mathcal{O}(2nr)$	$\mathcal{O}(2nr)$	1	✗	✗

Table 2: DeBERTaV3-base fine-tuning on GLUE. We compare with full fine-tuning (Full FT), Houlsby adapter (Houlsby et al., 2019) (HAdapter), Pfeiffer adapter (Pfeiffer et al., 2021) (PAdapter), LoRA (Hu et al., 2021), AdaLoRA (Zhang et al., 2023), DoRA (Mao et al., 2024), LoRA+(Hayou et al., 2024), and Bitfit(Zaken et al., 2022). We report target metrics and computational performance (higher is better) for the median of 5 runs using different random seeds. Best results per dataset are shown in bold. Results for BitFit, HAdapter, PAdapter were taken from (Zhang et al., 2023).

Method (# Params)	MNLI (Acc)	SST-2 (Acc)	CoLA (Mcc)	QQP (F1)	QNLI (Acc)	RTE (Acc)	MRPC (Acc)	STS-B (Corr)	Mean
Full FT (184M)	89.90	95.63	69.19	89.80	94.03	83.75	89.46	91.60	87.92
BitFit (0.1M)	89.37	94.84	66.96	84.95	92.24	78.70	87.75	91.35	85.77
HAdapter (1.22M)	90.13	95.53	68.64	89.27	94.11	84.48	89.95	91.48	87.94
PAdapter (1.18M)	90.33	95.61	68.77	89.40	**94.29**	85.20	89.46	91.54	88.07
LoRA r=8 (1.33M)	90.29	95.29	68.57	90.61	93.91	85.50	89.75	89.10	87.87
LoRA+ r=8 (1.33M)	90.39	95.367	70.56	90.65	94.05	85.59	89.65	88.07	88.04
DoRA r=8 (1.33M)	90.11	94.30	68.50	**90.71**	94.31	85.05	89.32	91.38	87.96
AdaLoRA target r=8 (1.27M)	**90.44**	95.64	68.76	90.65	94.11	**86.00**	89.44	91.41	88.30
AdaLoRA, matched parameters to GeoLoRA	90.21 (0.75M)	95.64 (1.27M)	68.59 (1.07M)	90.48 (0.72M)	93.93 (0.72M)	85.92 (1.16M)	88.21 (0.74M)	90.91(0.74M)	88.28 (0.89M)
GeoLoRA	90.38 (0.7M)	**95.98** (1.17M)	**69.03** (0.98M)	90.53 (0.69M)	94.23 (0.70M)	85.93 (1.19M)	**90.10** (0.75M)	**91.58** (0.71M)	**88.47** (0.86M)
Evaluation and train time comparison									
AdaLoRA (eval/train) [it/sec]	12.4/4.3	17.6/6.7	24.6/8.1	9.2/3.2	4.9/1.6	10.3/3.2	9,9/3.1	21.1/**8.5**	13.75/4.83
GeoLoRA (eval/train) [it/sec]	**17.1/4.9**	**21.3/8.3**	**37.4/9.1**	**12.0/3.8**	**5.9/1.8**	**13.2/3.7**	**12.6/3.7**	**21.3**/8.3	**17.6/5.6**

Performance analysis. The proposed method from Algorithm 1 combines low-rank optimality guarantees with computational efficiency gains compared to existing low-rank optimization methods, as shown in Table 1. For a rank r adapter, the computational cost of gradient evaluation (i.e., eq. (6)) is equivalent to that of AdaLoRA, which updates U, S, and V directly, and is similar to a standard LoRA update. The cost of basis augmentation is $\mathcal{O}(nr^2)$ due to the QR decomposition in eq. (7), comparable to evaluating the orthonormality regularization terms in AdaLoRA. Rank truncation is performed via an SVD of S at a cost of $\mathcal{O}(r^3)$, where typically $r \ll n$. The complexity analysis shows comparable per-iteration costs for LoRA, AdaLoRA, and GeoLoRA. In Table 2, we also report the number of iterations computed per second during training and evaluation for both GeoLoRA and AdaLoRA, demonstrating that GeoLoRA outperforms AdaLoRA across almost all GLUE benchmarks. We note that training and inference speed depend on both layer ranks and sequence lengths, and the performance difference is less pronounced for benchmarks with longer sequences.

Vision transformer for object classification. We compare GeoLoRA and AdaLoRA on fine-tuning the Vit-base-patch16-224 Vision Transformer, pre-trained on the Imagenet-1k dataset, and fine-tuned on Cifar10, Cifar100, and Tiny-Imagenet. GeoLoRa "local" uses a layer-wise rank truncation, and "global" uses the same global rank budget as AdaLoRA. Details on implementation and hyperparameters are provided in Appendix B.2. Table 3 shows that GeoLoRA achieves higher validation accuracy than AdaLoRA, while using fewer trainable parameters.

Ablations. In Figure 2, we examine how the performance of GeoLoRA is influenced by the initial rank and learning rate. Figure 2(a, b) demonstrate that GeoLoRA dynamically recovers the intrinsic rank of the low-rank adaptation, regardless of the initial rank, highlighting the robustness of the method with respect to this hyperparameter. Notably, GeoLoRA can extend the adapter rank to full rank if necessary within logarithmic time, while truncating in constant time (in terms of optimization iterations). We provide a detailed discussion of the rank distribution across transformer layers in Appendix B.2. Similarly, Figure 2(c, d) show that GeoLoRA is less sensitive to learning rate variations compared to AdaLoRA.

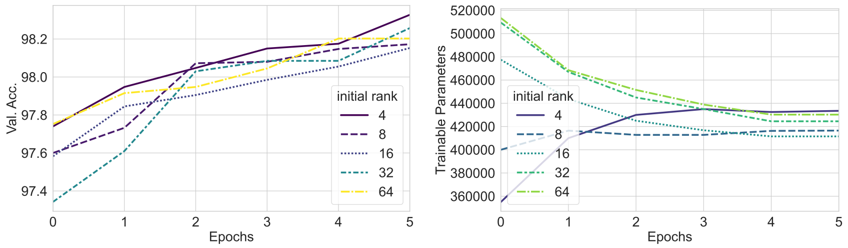
Dreambooth stable diffusion. Finally, we test GeoLoRA on fine-tuning Stable Diffusion (Rombach et al., 2021) using Dreambooth (Ruiz et al., 2023) on their original datasets. Implementation details are provided in Appendix B.3. In Table 4, we compare the validation loss and number of parameters

Table 3: Vit-base-patch16-224 fine-tuning on Cifar10, 100 and Tiny-Imagenet. We compare LoRa, AdaLoRA to GeoLoRA with local and global budgeting reporting the median of 5 runs using different random seeds. GeoLoRa "local" uses a layer-wise rank truncation, and "global" uses the same global rank budget as AdaLoRA.

Method	Cifar 10 [%]		Cifar 100 [%]		Tiny-Imagenet [%]	
	# Params	Acc [%]	# Params	Acc [%]	# Params	Acc [%]
LoRA	0.47M (r=3)	98.47	0.47M (r=3)	91.47	0.99M (r=6)	87.34
AdaLoRA	**0.47M**	98.51	0.45M	91.44	0.9M	87.21
GeoLoRA, local	**0.47M**	**98.55**	**0.35M**	**91.63**	0.92M	**88.09**
GeoLoRA, global	0.48M	98.51	0.47M	91.62	**0.75M**	88.07

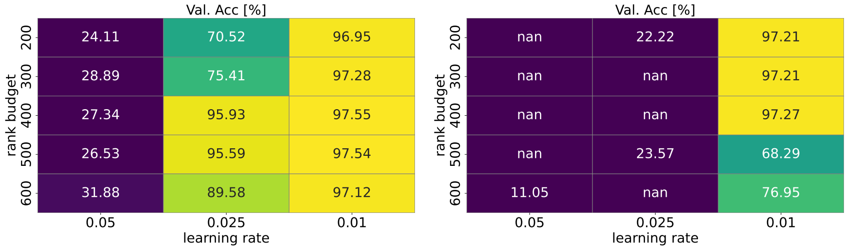
Table 4: Stable Diffusion on Dreambooth benenchmark. We compare LoRA and GeoLoRA reporting the median of 5 runs. r_0 for AdaLoRA is the initial rank, while r is the target rank.

Method	Val. Loss	# Params
LoRA ($r = 5$)	0.275	3.0 M
LoRA ($r = 3$)	0.281	1.8 M
AdaLoRA ($r_0 = 8, r = 5$)	0.245	4.7M
AdaLoRA ($r_0 = 8, r = 3$)	0.247	1.78M
GeoLoRA ($\tau = 0.02$)	**0.242**	2.6M
GeoLoRA ($\tau = 0.1$)	0.257	**1.4M**



(a) Validation accuracy over epochs.

(b) Trainable parameters over epochs.



(c) GeoLoRA

(d) AdaLoRA

Figure 2: **Top panels (a, b):** GeoLoRA-adapted ViT-32b fine-tuned on Cifar10 with different initial layer ranks, using a learning rate of $1e{-}3$ and $\tau = 0.3$. The total number of trainable parameters converges to a similar steady state, regardless of the initial rank. The differences in validation accuracy between runs are smaller than the variance observed within individual setups. **Bottom panels (c, d):** GeoLoRA- and AdaLoRA-adapted ViT-32b fine-tuned on Cifar10 with different rank budgets and learning rates. Fields marked with `nan` indicate that training diverged within the first epoch. GeoLoRA demonstrates significantly greater robustness than AdaLoRA, particularly with high learning rates.

between LoRA and GeoLoRA, showing that GeoLoRA consistently achieves lower validation loss with fewer trainable parameters.

6 CONCLUSION

We introduced GeoLoRA (Geometric Low-Rank Adaptation), a novel adaptive low-rank fine-tuning method that combines computational efficiency with robustness. Based on geometric principles from dynamical low-rank approximation theory, the method comes with guarantees of convergence and local optimality. By leveraging a parallel update strategy of the low-rank adapters, the method requires only a single backward pass per iteration, achieving inference and training speed comparable or superior to existing baselines such as AdaLoRA, and much more efficient than previous geometric-aware strategies. Our experiments on the GLUE benchmark, Vision Transformers, and Stable Diffusion demonstrate that GeoLoRA outperforms existing PEFT methods in both accuracy and efficiency, with fewer trainable parameters. These results, alongside strong theoretical guarantees, position GeoLoRA as a robust solution for efficient model adaptation.

REFERENCES

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning, 2020.

Gianluca Ceruti, Jonas Kusch, and Christian Lubich. A rank-adaptive robust integrator for dynamical low-rank approximation. *BIT Numerical Mathematics*, 2022. URL `https://doi.org/10.1007/s10543-021-00907-7`.

Gianluca Ceruti, Jonas Kusch, and Christian Lubich. A parallel rank-adaptive integrator for dynamical low-rank approximation, 2023. URL `https://arxiv.org/abs/2304.05660`.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv:1602.02830*, 2016.

Mehrdad Ghadiri, Matthew Fahrbach, Gang Fu, and Vahab Mirrokni. Approximately optimal core shapes for tensor decompositions. In *International Conference on Machine Learning*, pp. 11237–11254. PMLR, 2023.

Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. *Advances in neural information processing systems*, 29, 2016.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models, 2024.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing, 2023. URL `https://arxiv.org/abs/2111.09543`.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 1389–1397, 2017.

Arsen Hnatiuk, Jonas Kusch, Lisa Kusch, Nicolas R. Gauger, and Andrea Walther. Stochastic aspects of dynamical low-rank approximation in the context of machine learning. *Optimization Online*, 2024. doi: https://optimization-online.org/?p=25971. URL `https://optimization-online.org/?p=25971`.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/houlsby19a.html`.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Yerlan Idelbayev and Miguel A. Carreira-Perpinan. Low-rank compression of neural nets: Learning the rank of each layer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Mikhail Khodak, Neil Tenenholtz, Lester Mackey, and Nicolo Fusi. Initialization and regularization of factorized neural layers. In *International Conference on Learning Representations*, 2021.

O. Koch and C. Lubich. Dynamical low-rank approximation. *SIAM Journal on Matrix Analysis and Applications*, 29(2):434–454, 2007a. ISSN 0895-4798. doi: 10.1137/050639703. URL `https://doi.org/10.1137/050639703`.

Othmar Koch and Christian Lubich. Dynamical low-rank approximation. *SIAM Journal on Matrix Analysis and Applications*, 29(2):434–454, 2007b.

Vladislav Lialin, Namrata Shivagunde, Sherin Muckatira, and Anna Rumshisky. Relora: High-rank training through low-rank updates, 2023.

Yulong Mao, Kaiyu Huang, Changhao Guan, Ganglin Bao, Fengran Mo, and Jinan Xu. Dora: Enhancing parameter-efficient fine-tuning with dynamic rank distribution. 2024. URL `https://api.semanticscholar.org/CorpusID:270062642`.

P Molchanov, S Tyree, T Karras, T Aila, and J Kautz. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations*, 2017.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning, 2021. URL `https://arxiv.org/abs/2005.00247`.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

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, 2023. URL `https://arxiv.org/abs/2208.12242`.

Hiroyuki Sato. *Riemannian optimization and its applications*, volume 670. Springer, 2021.

Steffen Schotthöfer, Emanuele Zangrando, Jonas Kusch, Gianluca Ceruti, and Francesco Tudisco. Low-rank lottery tickets: finding efficient low-rank neural networks via matrix differential equations. In *Advances in Neural Information Processing Systems*, 2022. URL `https://proceedings.neurips.cc/paper_files/paper/2022/file/7e98b00eeafcdaeb0c5661fb9355be3a-Paper-Conference.pdf`.

Steffen Schotthöfer and M. Paul Laiu. Federated dynamical low-rank training with global loss convergence guarantees, 2024. URL `https://arxiv.org/abs/2406.17887`.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation, 2023.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2019. URL `https://arxiv.org/abs/1804.07461`.

Hongyi Wang, Saurabh Agarwal, and Dimitris Papailiopoulos. Pufferfish: Communication-efficient models at no extra cost. *Proceedings of Machine Learning and Systems*, 3:365–386, 2021.

Gerhard Wanner and Ernst Hairer. *Solving ordinary differential equations II*, volume 375. Springer Berlin Heidelberg New York, 1996.

Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4820–4828, 2016.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models, 2022. URL `https://arxiv.org/abs/2106.10199`.

Emanuele Zangrando, Steffen Schotthöfer, Gianluca Ceruti, Jonas Kusch, and Francesco Tudisco. Rank-adaptive spectral pruning of convolutional layers during training. In *Advances in Neural Information Processing Systems*, 2024.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning, 2023.

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

Algorithm 2: Various auxiliary functions

```
def optimizer_step($P$: param, $G$: gradient, $\lambda$: learning rate):
    $P^{\text{new}} \leftarrow P - \lambda G$            /* May use momentum and weight decay */
    return $P^{\text{new}}$
def basis_augmentation($B$: old basis, $G_B$: basis dynamics):
    $[B \mid \widetilde{B}] \leftarrow \texttt{qr}([B \mid G_B])$
    return $\widetilde{B}$
def truncation($\widehat{S}$: augmented coefficient, $\widehat{U}$: augmented basis, $\widehat{V}$: augmented co-basis ):
    $P_{r_1}, \Sigma_{r_1}, Q_{r_1} \leftarrow$ truncated $\texttt{svd}(\widetilde{S})$ with threshold $\vartheta$ to new rank $r_1$
    $U \leftarrow \widehat{U} P_{r_1}; V \leftarrow \widehat{V} Q_{r_1}$                          /* Basis update */
    $S \leftarrow \Sigma_{r_1}; S^{\text{inv}} \leftarrow \Sigma_{r_1}^{-1}$      /* Coefficient update with diagonal $\Sigma_{r_1}$ */
    return $U, S, V, S^{\text{inv}}$
```

A ALGORITHM FOR AUXILIARY FUNCTIONS

We present the auxiliary function for Algorithm 1 in Algorithm 2.

B ADDITIONAL INFORMATION FOR THE NUMERICAL TEST CASES

B.1 GLUE BENCHMARK

B.1.1 DATASET DESCRIPTION

We compare GeoLoRA to several fine-tuning methods from recent literature in the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019). The GLUE benchmark is a collection of diverse natural language understanding tasks designed to evaluate the performance of models in comprehending and processing human language. GLUE provides a comprehensive assessment by including tasks that cover a range of linguistic phenomena, such as textual entailment, sentiment analysis, sentence similarity, and more. The benchmark consists of nine different tasks:

- CoLA (Corpus of Linguistic Acceptability): Classifying whether a sentence is grammatically correct or not.
- SST-2 (Stanford Sentiment Treebank): Sentiment analysis task where the goal is to classify the sentiment of a sentence as positive or negative.
- MRPC (Microsoft Research Paraphrase Corpus): Identifying if two sentences are paraphrases of each other.
- STS-B (Semantic Textual Similarity Benchmark): Measuring the degree of semantic similarity between two sentences on a scale from 1 to 5.
- QQP (Quora Question Pairs): Determining if a pair of questions are semantically equivalent.
- MNLI (Multi-Genre Natural Language Inference): Classifying the relationship between a pair of sentences (entailment, contradiction, or neutral).
- QNLI (Question Natural Language Inference): Determining if a sentence provides a correct answer to a given question.
- RTE (Recognizing Textual Entailment): Binary classification task for entailment and contradiction.
- WNLI (Winograd Schema Challenge): Resolving pronoun reference ambiguity in sentences. Specific Focus: MRPC (Microsoft Research Paraphrase Corpus)

We present the benchmark overview in Table 5. To recapitulate, the F1 score is defined in dependence of precision score P and recall score R. The model precision P is given by

$$P := \frac{P_T}{P_T + P_F}, \tag{15}$$

Table 5: Summary of GLUE benchmark tasks

Corpus	**Task**	**#Train**	**#Dev**	**#Test**	**#Label**	**Metrics**
			Single-Sentence Classification (GLUE)			
CoLA	Acceptability	8.5k	1k	1k	2	Matthews corr
SST	Sentiment	67k	872	1.8k	2	Accuracy
			Pairwise Text Classification (GLUE)			
MNLI	NLI	393k	20k	20k	3	Accuracy
RTE	NLI	2.5k	276	3k	2	Accuracy
QQP	Paraphrase	364k	40k	391k	2	F1
MRPC	Paraphrase	3.7k	408	1.7k	2	Accuracy
QNLI	QA/NLI	108k	5.7k	5.7k	2	Accuracy
Text Similarity (GLUE)						
STS-B	Similarity	7k	1.5k	1.4k	1	Pearson/Spearman cor

where P_T is the number of true positive and P_F is the number of false positive examples. The recall R is the ratio

$$R := \frac{P_T}{P_T + N_F}, \tag{16}$$

where N_F are the false negatives. The F1 score combines these two metrics to

$$F1 := \frac{2PR}{P + R}. \tag{17}$$

B.1.2 REFERENCE IMPLEMENTATIONS

Full finetuning (FT): This is the most common approach for model finetuning and transfer learning. Here, the model is initialized with pre-trained weights and all model parameters are updated with gradient descent.

Bitfit (Zaken et al., 2022): Here, the model is initialized with pre-trained weights, but only bias terms are updated with gradient descent.

Adapter tuning (Houlsby et al., 2019; Pfeiffer et al., 2021): Two-layer adapters are inserted between transformer blocks. In (Houlsby et al., 2019), the adapter is inserted between the self-attention module and the feed-forward module and equipped with a residual connection. In (Pfeiffer et al., 2021), the adapter is applied after the feed-forward module and the layer-norm module. To maintain conistency with the notation of (Zhang et al., 2023), we call the method of (Houlsby et al., 2019) HAdapter and the method of (Pfeiffer et al., 2021) PAdapter.

LoRA (Hu et al., 2021): As stated in Section 3, LoRA applies additive corrections to selected weight matrices, i.e. $\mathbf{z} = \sigma(W_{\text{pt}}\mathbf{x} + \frac{\alpha}{r}AB^\top\mathbf{x})$ for $A, B \in \mathbb{R}^{n\times r}$. We apply LoRA to key W_k, query W_q and value W_v matrices of all attention blocks, and to both feed-forward layers W_{f_1} and W_{f_2}.

AdaLoRA (Zhang et al., 2023): As stated in Section 3, AdaLoRA applies additive corrections to selected weight matrices, i.e. $\mathbf{z} = \sigma(W_{\text{pt}}\mathbf{x} + \frac{\alpha}{r}USV^\top\mathbf{x})$ with arbitrary activation σ, frozen pre-trained weights $W_{\text{pt}} \in \mathbb{R}^{n\times n}$, rank r adapter weights $U, V \in \mathbb{R}^{n\times r}$, $S \in \mathbb{R}^{r\times r}$. An SVD-based truncation mechanism is used to select layer ranks. Alternatively, the loss-sensitivity of singular vectors can be used for layer rank selection. Just like LoRA, we apply AdaLoRA to key W_k, query W_q and value W_v matrices of all attention blocks, and to both feed-forward layers W_{f_1} and W_{f_2}.

We use the implementation of (Zhang et al., 2023, Appendix C) to compute the results for the presented reference methods and use their reported hyper-parameter choices: We compare the baselines under different budget levels, for example, given the total trainable parameters as 0.3/0.6/1.2 million. In order to match the parameter budget, we select the hidden dimensions of adapters from {8, 16, 32, 64}, set the rank r of LoRA as {2, 4, 8}, and choose the final budget $b^{(T)}$ of AdaLoRA from {144, 288, 576}. Then we set $b^{(0)}$ as 1.5 times of $b^{(T)}$ for AdaLoRA and select the regularization coefficient γ from $\{0.1, 0.3, 0.5\}$. We set the exponential moving average parameters β_1 and β_2 of AdamW as their default value 0.85. We select the learning rate from $\{5\text{e}{-5}, 8\text{e}{-5}, 1\text{e}{-4}, 2\text{e}{-4}\}$.

B.1.3 IMPLEMENTATION DETAILS

We implement GeoLoRA as similar as possible as Adalora to achieve a fair comparison. That is, we add an adapter of the form $\mathbf{z} = \sigma(W_{\mathrm{pt}}\mathbf{x} + USV^\top\mathbf{x})$ to the key W_k, query W_q and value W_v matrices of all attention blocks, and to both feed-forward layers W_{f_1} and W_{f_2}. For each adapter, we employ Algorithm 1 to update the layer weights and ranks.

In Table 6, we display the hyper-parameter choices of GeoLoRA

Table 6: Hyper-parameter setup for the GLUE benchmark.

Dataset	**Learning Rate**	**Batch Size**	**# Epochs**	τ	inital rank
MNLI	5×10^{-4}	32	7	0.15	10
RTE	1.2×10^{-3}	32	50	0.15	10
QNLI	1.2×10^{-3}	32	5	0.15	10
MRPC	1×10^{-3}	32	30	0.15	10
QQP	5×10^{-4}	32	5	0.15	10
SST-2	8×10^{-4}	32	24	0.15	10
CoLA	5×10^{-4}	32	25	0.15	10
STS-B	2.2×10^{-3}	32	25	0.15	10

B.2 OBJECT CLASSIFICATION BENCHMARKS FOR THE VIT-BASE-PATCH16-224 VISION TRANSFORMER

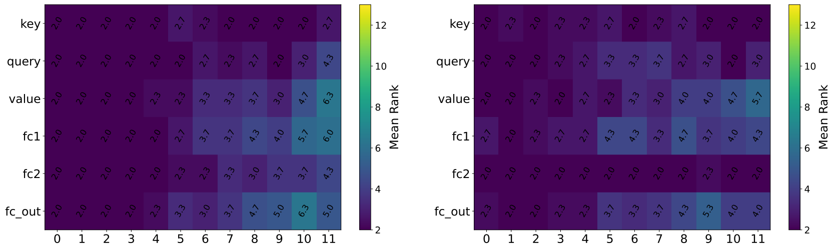
We present in Table 3 results for finetuning the vit-base-patch16-224 vision transformer, which is pretrained on the imagenet-1k-dataset. The pretrained weights are downloaded from the torch-vision python package. For both AdaLora and GeoLoRA, we augment the key, query, and value matrices from attention layers as well as the three fully connected layers of each transformer block with a low-rank adapter. The biases of each layer are trainable. Additionally, the classifier is augmented with a low-rank adapter. The classifier is low-rank by construction, and we fix the rank as the number of classes. We fine-tune the vision transformer on Cifar10, Cifar100 and Tiny-Imagenet.

The hyperparameter settings to generate the results of Table 3, Figure 3 and Figure 4 are given in Table 7.

Table 7: Hyper-parameter setup for fine-tuning vit-base-patch16-224 vision transformer with GeoLoRA.

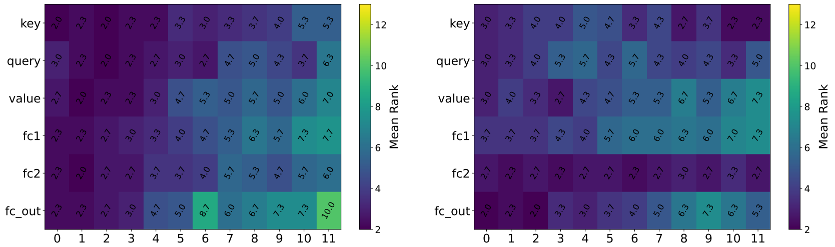
Dataset	**Learning Rate**	**Batch Size**	**# Epochs**	τ	inital rank
Cifar10	1×10^{-3}	256	5	0.2	16
Cifar100	1×10^{-3}	256	5	0.25	32
TinyImageNet	1×10^{-4}	256	5	0.15	32

Figure 3 and Figure 4 show the rank distribution across layers for both AdaLoRA and GeoLoRA with global budget, for learning rate $\lambda = 1\mathrm{e}{-3}$ and $\lambda = 1\mathrm{e}{-4}$ and budgets ranging from $b = 200, \dots, 600$ total ranks for the network. Both methods prefer to allocate higher ranks to the deeper layers of the vision transformer, and prefer fully-connected layers over attention layers. Both methods pefer the first fully connected layer of a transformer block over the second. Overall GeoLoRA tends to assert higher ranks to single layers, compared to AdaLora, that distributes ranks more heterogeneously. The effects are more pronounced for smaller learning rates.



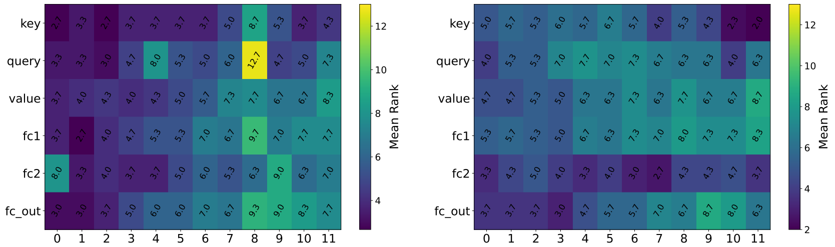
(a) GeoLoRA: mean b=200

(b) AdaLoRA: mean b=200



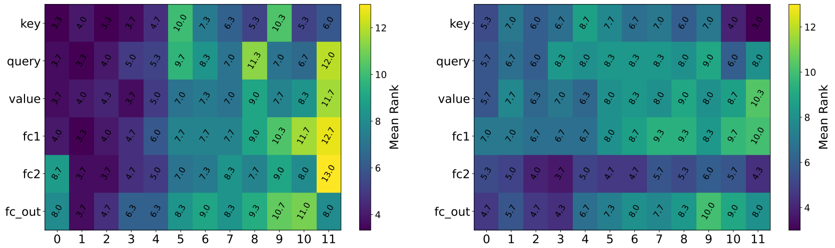
(c) GeoLoRA: mean b=300

(d) AdaLoRA: mean b=300



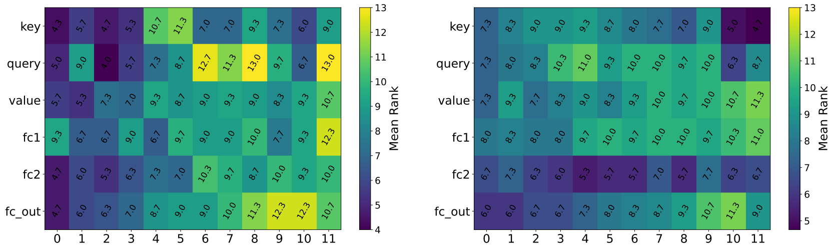
(e) GeoLoRA: mean b=400

(f) AdaLoRA: mean b=400



(g) GeoLoRA: mean b=500

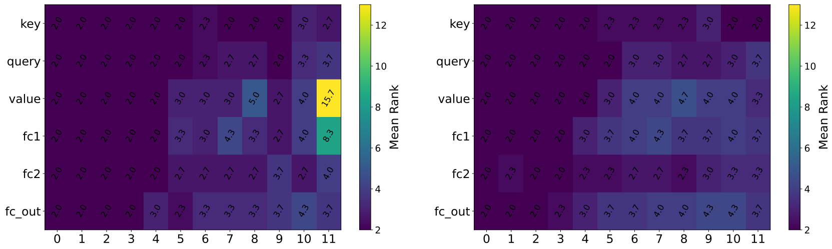
(h) AdaLoRA: mean b=500



(i) GeoLoRA: mean b=600

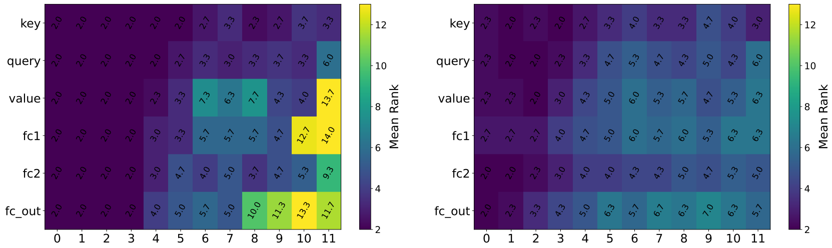
(j) AdaLoRA: mean b=600

Figure 3: Rank distribution of Vit-32b finetuned on Cifar10 for 5 epochs at learning rate $1e{-}3$ using GeoLoRA and AdaLoRA.



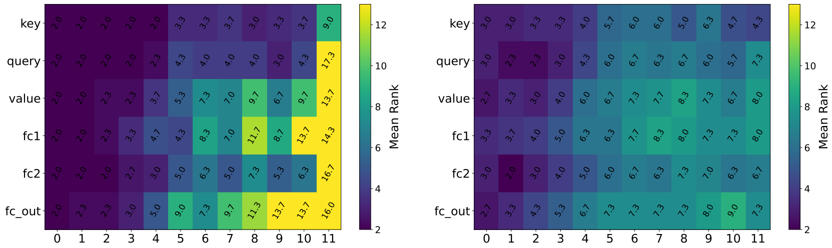
(a) GeoLoRA: mean b=200

(b) AdaLoRA: mean b=200



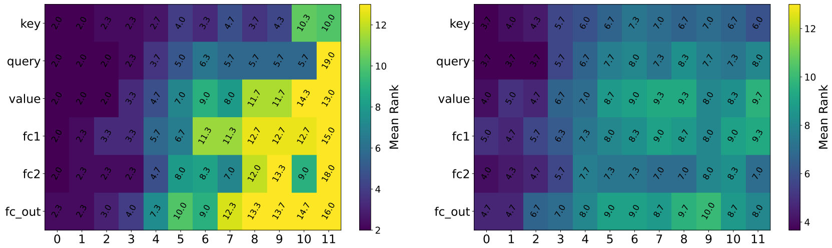
(c) GeoLoRA: mean b=300

(d) AdaLoRA: mean b=300



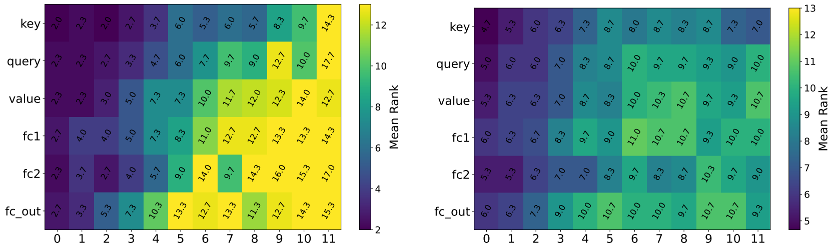
(e) GeoLoRA: mean b=400

(f) AdaLoRA: mean b=400



(g) GeoLoRA: mean b=500

(h) AdaLoRA: mean b=500



(i) GeoLoRA: mean b=600

(j) AdaLoRA: mean b=600

Figure 4: Rank distribution of Vit-32b finetuned on Cifar10 for 5 epochs at learning rate $1\mathrm{e}{-4}$ using GeoLoRA and AdaLoRA.

B.3 STABLE DIFFUSION ON DREAMBOOTH,

In this numerical example, we apply low-rank adapters to all linear and attention layers of the U-Net and the text encoder networks. The hyperparameters for LoRA and GeoLoRA are the same, apart from the fact that we start with adapters of rank 8 for both Unet and text encoder. We train for 5 full epochs, using adamW as an optimizer, with $(\beta_1, \beta_2) = (0.9, 0.999)$, initial learning rate 5×10^{-6} and weight decay set to 10^{-2}.

C OVERVIEW FOR THE NUMERICAL ANALYSIS

C.1 NOTATION

We provide an overview of the notation used throughout the main manuscript and the appendix.

- $W \in \mathbb{R}^{n\times n}$ is the full-rank weight matrix of a neural network layer or adapter.
- $Z \in \mathbb{R}^{n\times n}$ is an arbitrary matrix.
- $\mathcal{M}_r = \{Z \in \mathbb{R}^{n\times n} : \text{rank}(Z) = r\}$ is a manifold of rank r matrices.
- $\mathcal{T}_Z\mathcal{M}_r$ is the tangent space of $\mathcal{M}_r$ at Z for any $Z \in \mathbb{R}^{n\times n}$.
- $W^r = USV^\top \in \mathcal{M}_r$ is a rank-r approximation of a matrix W.
- $\widehat{W}^r = \widehat{U}\widehat{S}\widehat{V}^\top \in \mathcal{M}_r$ is a rank-r approximation of a matrix W with augmented basis.
- $U, V \in \mathbb{R}^{n\times r}$ is the orthonormal basis and co-basis $\mathcal{M}_r$.
- $\widehat{U} = [U_0, \widetilde{U}] \in \mathbb{R}n \times 2r$ is the augmented basis. (Analogously for $\widehat{V}$.)
- $U_0 \in \mathbb{R}^{n\times r}$ is the basis at the beginning of the iteration. (Analogously for V_0.)
- $\widetilde{U} \in \mathbb{R}^{n\times r}$ is the basis augmentation, obtained by $[U_0, \widetilde{U}] = \text{ortho}([U_0, K^{\text{new}}]) \in \mathbb{R}^{n\times 2r_0}$. (Analogously for $\widetilde{V}$.)
- $S \in \mathbb{R}^{r\times r}$ is the coefficient matrix so assemble the low-rank approximation W^r from U, V.
- $P(Z)$ is the orthogonal projection onto $\mathcal{T}_Z\mathcal{M}_r$.
- $P_U = UU^\top$ is the orthogonal projection onto the range of orthonormal $U \in \mathbb{R}^{n\times r}$.
- $P_V = VV^\top$ is the orthogonal projection onto the range of orthonormal $V \in \mathbb{R}^{n\times r}$.
- When applied to vectors, $\|\cdot\|$ denotes the Euclidean norm (ℓ_2-norm). When applied to matrices, $\|\cdot\|$ denotes the Frobenius norm.
- $\mathcal{L}(W;\xi)$ denotes the loss function dependent on weight matrix W and data sample randomness ξ. Commonly abbreviated by $\mathcal{L}(W)$.
- $f(W;\xi) = -\nabla_W\mathcal{L}(W;\xi)$ is the negative stochastic loss gradient w.r.t W. Commonly abbreviated by $f(W)$.
- $F(W) = \mathbb{E}_\xi[f(W,\xi)]$ the expectation of the random loss gradient, called the deterministic gradient.

C.2 RECAP OF COMMONLY USED PROPERTIES

We recapitulate repeatedly used properties of the mathematical objects and notations introduced above.

- Per definition, we have for any $Z \in \mathbb{R}^{n\times n}$
$$P(W^r)Z = UU^\top Z + ZVV^\top - UU^\top ZVV^\top \tag{18}$$
- Since $\mathcal{T}_Z\mathcal{M}_r$ is a subspace of $\mathbb{R}^{n\times n}$ for any $Z \in \mathcal{M}_r$ we can decompose the gradients F and f into $F(Z) = M(Z) + R(Z)$ and $f(Z) = f(Z) + r(Z)$ for any $Z \in \mathcal{R}^{n\times n}$, where $M(Z), m(Z) \in \mathcal{T}_Z\mathcal{M}_r$.

C.3 GLOBAL ASSUMPTIONS

The following provides a comprehensive overview of the global assumptions for made in the analysis section and proofs of the provided theorems. The assumptions are common in literature, see e.g. (Hnatiuk et al., 2024)

Assumption 1. *There is an* $\varepsilon > 0$ *such that* $\|R(Z)\|, \|r(Z)\| \leq \epsilon$ *for all* $Z \in \mathcal{M}_r$

Assumption 2. F *and* f *are bounded by a constant* $B > 0$ *and L-continuous w.r.t.* $\|\cdot\|$ *with constant* $L > 0$.

Assumption 3. *There is a constant* $C > 0$ *such that* $\|F(Z) - f(Z)\| \leq C$.

Assumption 4. *At initial time, we assume that the difference of a full-rank weight matrix* W_0 *and its low-rank counterpart* W_0^r *is bounded by* $\|W_0 - Y_0\| \leq \delta$ *for* $\delta > 0$.

Assumption 5. *For all times, we have w.l.o.g* $\mathcal{L}(t) > 0$.

D DESCENT DIRECTION

We first state a few auxiliary lemmas, which provide common inequalities that will be used in the following analysis.

Lemma 1. *(Hnatiuk et al., 2024, Lemma 5.2) For any two matrices* $Y_1, Y_2 \in \mathbb{R}^{n\times n}$ *and an* L*-smooth* $\mathcal{L}$ *with constant* L *it holds*

$$\mathcal{L}(Y_1) - \mathcal{L}(Y_2) \leq -\langle Y_1 - Y_2, f(Y_2)\rangle + \frac{L}{2}\left\|Y_1 - Y_2\right\|^2, \tag{19}$$

where $f(Y) = -\nabla_Y \mathcal{L}(Y)$*. Furthermore, it holds*

$$\mathcal{L}(Y_1) - \mathcal{L}(Y_2) \leq -\langle Y_1 - Y_2, F(Y_2)\rangle + \frac{L}{2}\left\|Y_1 - Y_2\right\|^2, \tag{20}$$

where $F(Y) = -\mathbb{E}[\nabla_Y \mathcal{L}(Y)]$*.*

The following results are primarily based on (Hnatiuk et al., 2024) and use the reformulation of truncated terms as proposed in (Zangrando et al., 2024). For ease of notation, we use $f(W_t^r) = f(W_t^r, \xi_t)$. Hence, randomness is not explicitly stated in our notation. Note that in this case, the factorized solution $W_1^r = U_1 S_1 V_1^\top$ is random since it depends on $f(W_1^r)$. When using expected values, we explicitly write down the corresponding random variable. That is, $\mathbb{E}_\xi[\cdot]$ is the expected value for a random variable ξ. We denote the random variable in step T as ξ_T and denote $\mathbb{E}[\cdot] := \mathbb{E}_{\xi_1,\cdots,\xi_T}[\cdot]$.

Theorem 4. *(Restatement of Theorem 1) Algorithm 1 with stochastic (mini-batch) gradients fulfills*

$$\mathbb{E}_{\xi_{t+1}}[\mathcal{L}(W_{t+1}^r)] \leq \mathcal{L}(W_t^r) - \lambda\left(1 - \frac{L\lambda^2}{2}\right)\mathbb{E}_{\xi_1}[\|P(W_t^r)f(W_t^r,\xi_t)\|^2] + L\mathbb{E}_{\xi_1}[\|W_{t+1}^r - \widehat{W}_t^r\|]. \tag{21}$$

where W_t^r*,* $\widehat{W}_t^r$*,* W_{t+1}^r *are the low-rank weight matrices at the start of iteration* $t+1$*, before, and after the truncation step, respectively. The step size is given by* λ*.*

Proof. Without loss of generality, we restrict ourselves to time steps $t = 0$ and write $f(W_0^r)$ shorthand for $f(W_{t=0}^r, \xi_t)$. By definition of the coefficient matrix assembly in eq. (8), we get respectively

- $\widetilde{U}\widetilde{U}^\top f(W_0^r)V_0V_0^\top$ for the right hand side of the S^{new} block
- $U_0U_0^\top f(W_0^r)\widetilde{V}\widetilde{V}^\top$ for the right hand side of the L^{new} block
- $\widetilde{U}\widetilde{U}^\top f(W_0^r)V_0V_0^\top$ for the right hand side of the K^{new} block
- and zero for the lower right block.

Since the augmented bases are orthonormal, we can write for $W_0^r = U_0S_0V_0$

$$\begin{aligned}
\widehat{W}_0^r &\overset{(8)}{=} W_0^r + \lambda U_0U_0^\top f(W_0^r)V_0V_0^\top + \lambda\widetilde{U}\widetilde{U}^\top f(W_0^r)V_0V_0^\top + \lambda U_0U_0^\top f(W_0^r)\widetilde{V}\widetilde{V}^\top \\
&= W_0^r - \lambda U_0U_0^\top f(W_0^r)V_0V_0^\top + \lambda\widehat{U}\widehat{U}^\top f(W_0^r)V_0V_0^\top + \lambda U_0U_0^\top f(W_0^r)\widehat{V}\widehat{V}^\top \\
&= W_0^r - \lambda U_0U_0^\top f(W_0^r)V_0V_0^\top + \lambda f(W_0^r)V_0V_0^\top + \lambda U_0U_0^\top f(W_0^r) \\
&\overset{(18)}{=} W_0^r + \lambda P(W_0^r)f(W_0^r).
\end{aligned}$$

By Lemma 1 we have

$$\mathcal{L}(\widehat{W}_0^r) - \mathcal{L}(W_0^r) \leq -\langle f(W_0^r), \widehat{W}_0^r - W_0^r\rangle + \frac{L}{2}\|\widehat{W}_0^r - W_0^r\|^2. \tag{22}$$

Therefore, plugging the above equation into eq. (22) yields

$$\begin{aligned}
\mathcal{L}(\widehat{W}_0^r) - \mathcal{L}(W_0^r) \leq& -\lambda\langle f(W_0^r), P(W_0^r)f(W_0^r)\rangle + \frac{L\lambda^2}{2}\|P(W_0^r)f(W_0^r)\|^2 && (23)\\
=& -\lambda\langle P(W_0^r)f(W_0^r), P(W_0^r)f(W_0^r)\rangle + \frac{L\lambda^2}{2}\|P(W_0^r)f(W_0^r)\|^2 && (24)\\
=& -\lambda\left(1 - \frac{L\lambda^2}{2}\right)\|P(W_0^r)f(W_0^r)\|^2. && (25)
\end{aligned}$$

where the second line is obtained by definition of the orthogonal projection. Comparing the loss before $\widehat{W}^r$ and after W_1^r truncation yields for some $s \in (0,1)$ using the mean value theorem and the Cauchy-Schwarz inequality,

$$\mathcal{L}(W_1^r) \leq \mathcal{L}(\widehat{W}_0^r + \langle \nabla \mathcal{L}(sW_1^r + (1-s)\widehat{W}^r), W_1^r - \widehat{W}_0^r \rangle \leq \mathcal{L}(\widehat{W}_0^r) + L\|W_1^r - \widehat{W}_0^r\|. \tag{26}$$

Plugging eq. (26) into eq. (23) then gives

$$\mathcal{L}(W_1^r) - \mathcal{L}(W_0^r) \leq -\lambda \left(1 - \frac{L\lambda^2}{2}\right) \|P(W_0^r)f(W_0^r)\|^2 + L\|W_1^r - \widehat{W}_0^r\|,$$

where L is the Lipschitz constant of F. Hence, taking the expected value yields

$$\mathbb{E}_{\xi_1}[\mathcal{L}(W_1^r)] \leq \mathcal{L}(W_0^r) - \lambda \left(1 - \frac{L\lambda^2}{2}\right) \mathbb{E}_{\xi_1}[\|P(W_0^r)f(W_0^r)\|^2] + L\mathbb{E}_{\xi_1}[\|W_1^r - \widehat{W}_0^r\|]\,.$$

□

E CONVERGENCE

Theorem 5. *(Restatement of Theorem 2) Let $\mathcal{L} \geq 0$ and $W_1^r, \ldots, W_T^r$ be the solutions generated by Algorithm 1 over T steps. Let the learning rate sequence $\{\lambda_t\}$ satisfy the Robbins-Monro conditions:*

$$\textstyle\sum_t \lambda_t = +\infty \qquad \sum_t \lambda_t^2 < +\infty\,.$$

Further assume $\sum_{t=1}^{T-1} \mathbb{E}[\|W_{t+1}^r - \widehat{W}_t^r\|] \leq D < \infty$, i.e. after some time, the solution W_t^r is contained in a manifold of rank r. Then we have

$$\liminf_{T\to\infty} \mathbb{E}[\|P(W_T^r)f(W_T^r)\|^2] = 0\,,$$

where the expected value is taken over all ξ_t.

Proof. By taking the expected value over $\xi_1, \ldots, \xi_T$ in eq. (21) and denoting the corresponding expected value as $\mathbb{E}[\cdot]$ we get

$$\begin{aligned}\mathbb{E}[\mathcal{L}(W_{t+1}^r)] - \mathbb{E}[\mathcal{L}(W_t^r)] \leq& -\lambda_t \mathbb{E}[\|P(W_t^r)f(W_t^r)\|^2] + \frac{L{\lambda_t}^2}{2}\mathbb{E}[\|P(W_t^r)f(W_t^r)\|^2] \\ &+ L\mathbb{E}[\|W_{t+1}^r - \widehat{W}_t^r\|] \\ =& -\lambda_t \left(1 - \frac{L\lambda_t}{2}\right) \mathbb{E}[\|P(W_t^r)f(W_t^r)\|^2] + L\mathbb{E}[\|W_{t+1}^r - \widehat{W}_t^r\|]\,.\end{aligned}$$

Using a telescoping sum until $t = T$ then yields

$$\begin{aligned}-\mathcal{L}(Y_0) \leq \mathbb{E}[\mathcal{L}(W_t^r)] - \mathcal{L}(Y_0) \leq& -\sum_{t=1}^{T-1} \lambda_t \left(1 - \frac{L\lambda_t}{2}\right) \mathbb{E}[\|P(W_t^r)f(W_t^r)\|^2] \\ &+ L\sum_{t=1}^{T-1} \mathbb{E}[\|W_{t+1}^r - \widehat{W}_t^r\|]\,.\end{aligned}$$

Rearranging gives

$$\begin{aligned}\sum_{t=1}^{T-1} \lambda_t \left(1 - \frac{L\lambda_t}{2}\right) \mathbb{E}[\|P(W_t^r)f(W_t^r)\|^2] &\leq \mathcal{L}(Y_0) + L\sum_{t=1}^{T-1} \mathbb{E}[\|W_{t+1}^r - \widehat{W}_{t+1}^r\|]\,. \\ &\leq \mathcal{L}(Y_0) + LD\,.\end{aligned}$$

Using the assumptions $\|P(W_t^r)f(W_t^r)\| \leq B$ and $\sum_{t=1}^{T-1} \mathbb{E}[\|W_{t+1}^r - \widehat{W}_{t+1}^r\|] \leq D$. Now, when $T \to \infty$, then the right-hand side remains bounded, implying that

$$\liminf_{T\to\infty} \mathbb{E}[\|P(W_t^r)f(W_t^r)\|^2] = 0\,.$$

□

F EFFICIENT EVALUATION OF THE RIGHT HAND SIDE OF THE LOW-RANK DYNAMICS

Algorithm 1 creates a trajectory in the low-rank parameter space, that robustly follows the full-rank solution of the gradient flow of the neural network training. In particular, Theorem 6 yields a time-continuous representation of Algorithm 1.

Theorem 6. *The evolution equations eq.* (6) *are explicit Euler discretizations of a dynamical system which is equivalent to*

$$\begin{aligned} \dot{S} &= -\nabla_S \mathcal{L}(U_0 S(t) V_0^\top), && S(t=0) = S_0, \\ \dot{K} &= -\nabla_K \mathcal{L}(K(t) V_0^\top), && K(t=0) = U_0 S_0, \\ \dot{L} &= -\nabla_L \mathcal{L}(U_0 L(t)^\top), && L(t=0) = S_0^\top V_0, \end{aligned} \tag{27}$$

where $\mathcal{L}$ is the stochastic loss given random data samples.

Proof. Consider the continuous time dynamics of $\dot{K}$, where we omit explicit time dependence on U, S, V and K for the sake of brevity, i.e.,

$$\begin{aligned} \dot{K} &= (\dot{US}) \\ &= \dot{U}S + U\dot{S} \\ &\overset{(4)}{=} -(I - UU^\top)\nabla_W \mathcal{L}(USV^\top)VS^{-1}S - UU^\top \nabla_W \mathcal{L}(USV^\top)V \\ &= -(I - UU^\top)\nabla_W \mathcal{L}(USV^\top)V - UU^\top \nabla_W \mathcal{L}(USV^\top)V \\ &= (UU^\top - I)\nabla_W \mathcal{L}(USV^\top)V - UU^\top \nabla_W \mathcal{L}(USV^\top)V \\ &= -\nabla_W \mathcal{L}(USV^\top)V \end{aligned} \tag{28}$$

Further, using the chain rule, we observe

$$\nabla_U \mathcal{L}(USV^\top) = \nabla_W \mathcal{L}(USV^\top)\nabla_U(USV^\top) = \nabla_W \mathcal{L}(USV^\top)VS^\top .$$

Thus, $-\nabla_U \mathcal{L}(USV^\top)S^{-\top} = -\nabla_W \mathcal{L}(USV^\top)V = \dot{K}$. Lastly we have by the chain rule $\dot{K} = -\nabla_W \mathcal{L}(USV^\top)V = -\nabla_K \mathcal{L}(USV^\top)$, which yields

$$\dot{K} = -\nabla_U \mathcal{L}(USV^\top)S^{-\top} = -\nabla_K \mathcal{L}(KV^\top) .$$

Analogously we obtain for $\dot{L}$

$$\dot{L} = -\nabla_V \mathcal{L}(USV^\top)S^{-1} = -\nabla_L \mathcal{L}(UL^\top) ,$$

which concludes the proof. □

Note that using an explicit Euler time discretization for eq. (27) directly yields eq. (6), the update step of GeoLoRA.

G ROBUST ERROR BOUND OF THE LOW-RANK SYSTEM

We show the robust error bound for Algorithm 1 applied to a single layer, and then extend the result to a network containing multiple layers treated with Algorithm 1.

Theorem 7. *(Restatement of Theorem 3) For an integer k, let $t = k\lambda$. Let $W(t)$ be the solution of eq.* (12)*, and let W_t^r be the factorized low-rank solution after k steps with Algorithm 1. Assume that for any Z in a neighborhood of $W(t)$, we have $\|(I - P(Z))\nabla\mathcal{L}(Z)\| < \varepsilon$, i.e., the gradient flow is close to $T_Z\mathcal{M}_r$. Then,*

$$\|W(t) - W_t^r\| \le c_1\varepsilon + c_2\lambda + c_3\vartheta/\lambda . \tag{29}$$

Moreover, let $W_{RF}(t)$ denote the solution of the Riemannian flow of equation 4. Then,

$$\|W_{RF}(t) - W_t^r\| \le c_4\varepsilon + c_2\lambda + c_3\vartheta/\lambda \tag{30}$$

where the constants c_1, c_2, c_3, c_4 depend only on L and B.

Proof. Let us first investigate the local error. That is, we choose the solution at a given time t_0 of the full-rank gradient flow of eq. (12), denoted as $W(t_0)$, as a given iteration of GeoLoRA, which we denote as W_0^r. Hence, $W(t_0) = W_0^r =: W_0 \in \mathcal{M}_r$. We are then interested in bounding the distance between the full-rank flow at $t_1 = t_0 + \lambda$ to the GeoLoRA solution after a single iteration with learning rate λ. To simplify notation, we denote $\widehat{U} = [U_0|\widetilde{U}] \in \mathbb{R}^{n\times 2r_0}$, $\widehat{V} = [V_0|\widetilde{V}] \in \mathbb{R}^{n\times 2r_0}$ and denote the projections onto these augmented basis vectors as $P_{\widehat{U}} = \widehat{U}\widehat{U}^\top$ and $P_{\widehat{V}} = \widehat{V}\widehat{V}^\top$. Moreover, c denotes a generic constant that only depends on L and B. It is important to note that this constant does not depend on S_k^{-1}, since we never perform Taylor expansions of the individual low-rank factors.

Let us denote the augmented solution of GeoLoRA before truncation as $\widehat{W}^r = \widehat{U}\widehat{S}\widehat{V}^\top$. Similarly, W_1^r is the truncated solution after iteration 1. Then, the local error is bounded by

$$\begin{aligned}\|W(t_1) - W_1^r\| \leq{}& \left\|W(t_1) - P_{\widehat{U}}W(t_1)P_{\widehat{V}}\right\| + \\ & \left\|P_{\widehat{U}}W(t_1)P_{\widehat{V}} - \widehat{W}^r\right\| + \left\|\widehat{W}^r - W_1^r\right\|.\end{aligned}$$

In the following, we bound the three norms individually in three corresponding steps.

Step 1 - Bounding $\left\|W(t_1) - P_{\widehat{U}}W(t_1)P_{\widehat{V}}\right\|$: Using the triangle inequality, we obtain

$$\begin{aligned}\left\|W(t_1) - P_{\widehat{U}}W(t_1)P_{\widehat{V}}\right\| &\leq \left\|W(t_1) - P_{\widehat{U}}W(t_1)\right\| + \left\|P_{\widehat{U}}W(t_1)(I - P_{\widehat{V}})\right\| \\ &= \left\|(I - P_{\widehat{U}})W(t_1)\right\| + \left\|W(t_1)(I - P_{\widehat{V}})\right\|,\end{aligned}$$

using orthonormality of $\widehat{U}$.

First term: Consider the first term with the dynamics $\dot{W}(t) = f(W)$ in mind,

$$\begin{aligned}&\left\|(I - P_{\widehat{U}})W(t_1)\right\| \\ \overset{\text{(I)}}{\leq}{}& \left\|(I - P_{\widehat{U}})(W_0 + \lambda f(W_0))\right\| + c\lambda^2 \\ \leq{}& \left\|(I - P_{\widehat{U}})(W_0 - \lambda P(W_0)f(W_0) + \lambda(I - P(W_0))f(W_0))\right\| + c\lambda^2 \\ \leq{}& \left\|(I - P_{\widehat{U}})W_0\right\| + \lambda\left\|(I - P_{\widehat{U}})P(W_0)f(W_0)\right\| + \lambda\left\|(I - P_{\widehat{U}})(I - P(W_0))f(W_0))\right\| + c\lambda^2 \\ \overset{\text{(II)}}{=}{}& \lambda\left\|(I - P_{\widehat{U}})P(W_0)f(W_0)\right\| + \lambda\left\|(I - P_{\widehat{U}})(I - P(W_0))f(W_0))\right\| + c\lambda^2 \\ \overset{\text{(III)}}{\leq}{}& \lambda\left\|(I - P_{\widehat{U}})P(W_0)f(W_0)\right\| + \lambda\varepsilon + c\lambda^2 \\ \overset{\text{(IV)}}{\leq}{}& \lambda\left\|(I - P_{\widehat{U}})f(W_0)\widehat{V}\widehat{V}^\top\right\| + \lambda\varepsilon + c\lambda^2.\end{aligned}$$

using Taylor expansion in (I), $W_0 \in \mathcal{M}_r$ in (II), Assumption 1 in (III), and eq. (18) in (IV).

By construction of the basis augmentation, we obtain

$$(I - P_{\widehat{U}})K^{\text{new}} = (I - P_{\widehat{U}})U_0S_0 = 0. \tag{31}$$

From eq. (31) we can directly conclude that $\left\|(I - P_{\widehat{U}})f(W_0)V_0V_0^\top\right\| = 0$. Thus we obtain

$$\begin{aligned}\lambda\left\|(I - P_{\widehat{U}})f(W_0)\widehat{V}\widehat{V}^\top\right\| &= \lambda\left\|(I - P_{\widehat{U}})f(W_0)V_0V_0^\top\right\| + \lambda\left\|(I - P_{\widehat{U}})f(W_0)\widetilde{V}\widetilde{V}^\top\right\| \\ &\leq \lambda\epsilon,\end{aligned}$$

where we used for the second term that $\widetilde{V}$ is in the orthogonal complement of V_0. Hence,

$$\left\|(I - P_{\widehat{U}})W(t_1)\right\| \leq c\lambda^2 + \lambda\varepsilon\,.$$

Second term: The same derivation for the co-range using the evolution for $L(t)$ yields

$$\left\|W(t_1)(I - P_{\widehat{V}})\right\| \leq c\lambda^2 + \lambda\varepsilon.$$

Step 2 - Bounding $\left\|P_{\widehat{U}}W(t_1)P_{\widehat{V}} - \widehat{W}^r\right\|$: We have by the assembly of the augmented S matrix in eq. (8),

$$\widehat{W}^r = \widehat{U}\widehat{S}\widehat{V}^\top = U_0S^{\text{new}}V_0^\top + \widetilde{U}\widetilde{U}^\top K^{\text{new}}V_0^\top + U_0L^{\text{new},\top}\widetilde{V}\widetilde{V}^\top,$$

from which we obtain the error bound between the projected $W(t_1)$ and $\widehat{W}^r$:

$$\begin{aligned}\left\| P_{\widehat{U}} W(t_1) P_{\widehat{V}} - \widehat{W}^r \right\| &\leq \left\| P_{\widehat{U}} W(t_1) P_{\widehat{V}} - U_0 S^{\mathrm{new}} V_0^\top + \widetilde{U}\widetilde{U}^\top K^{\mathrm{new}} V_0^\top + U_0 L^{\mathrm{new},\top} \widetilde{V}\widetilde{V}^\top \right\| \\ &\overset{(I)}{\leq} \left\| U_0^\top W(t_1) V_0 - S^{\mathrm{new}} \right\| + \left\| \widetilde{U}^\top W(t_1) V_0 - \widetilde{U}^\top K^{\mathrm{new}} \right\| \\ &\quad + \left\| U_0^\top W(t_1) \widetilde{V} - L^{\mathrm{new},\top} \widetilde{V} \right\| + \left\| \widetilde{U}^\top W(t_1) \widetilde{V} \right\| .\end{aligned}$$

where we use orthonormality of $\widehat{U}$,$\widehat{V}$ in (I). All terms on the right-hand side can be bounded by λ^2 and ε terms:

First term: We have

$$\begin{aligned}\left\| U_0^\top W(t_1) V_0 - S^{\mathrm{new}} \right\| &\overset{\text{(I)}}{=} \left\| \int_{t_0}^{t_1} U_0^\top (f(W(t)) - f(W_0)) V_0 \, dt \right\| \\ &\overset{\text{(II)}}{\leq} \int_{t_0}^{t_1} \left\| f(W(t)) - f(W_0) \right\| dt \\ &\overset{\text{(III)}}{=} \int_{t_0}^{t_1} \left\| f(W(t_0)) - f(W_0) \right\| dt + c\lambda^2 \\ &\overset{\text{(IV)}}{=} c\lambda^2\end{aligned}$$

where we use in (I) $S^{\mathrm{new}} = S_0 - U_0^\top \nabla_W \mathcal{L}(W_0; \xi) V_0 = -U_0^\top f(W_0) V_0$. We use the orthonormality of U_0, V_0 in (II), perform a Taylor expansion of the full-rank flow in (III), and finally use that $W(t_0) = W^S(t_0)$ in (IV).

Second and third term: We have

$$\begin{aligned}\left\| \widetilde{U}^\top W(t_1) V_0 - \widetilde{U}^\top K^{\mathrm{new}} \right\| &\overset{\text{(I)}}{\leq} \int_{t_0}^{t_1} \left\| \widetilde{U}^\top (f(W(t)) - f(W_0)) V_0 \right\| dt \\ &\overset{\text{(II)}}{\leq} \int_{t_0}^{t_1} \left\| f(W(t_0)) - f(W_0) \right\| dt + c\lambda^2 \\ &= c\lambda^2 ,\end{aligned}$$

where we use the K-step of GeoLoRA in (I) and a Taylor expansion of the full-rank flow in (II). $\left\| U_0^\top W(t_1) \widetilde{V} - L^{\mathrm{new},\top} \widetilde{V} \right\|$ can be bounded analogously.

Fourth term: Lastly, we obtain for the fourth term,

$$\begin{aligned}\left\| \widetilde{U}^\top W(t_1) \widetilde{V} \right\| &= \left\| \widetilde{U}^\top W(t_0) \widetilde{V} + \int_{t_0}^{t_1} \widetilde{U}^\top f(W(t)) \widetilde{V} \, dt \right\| \\ &\overset{\text{(I)}}{\leq} \int_{t_0}^{t_1} \left\| \widetilde{U}^\top f(W(t)) \widetilde{V} \right\| dt \\ &\leq \int_{t_0}^{t_1} \left\| \widetilde{U}^\top f(W(t_0)) \widetilde{V} \right\| dt + c\lambda^2 \overset{\text{(II)}}{\leq} \lambda\varepsilon + c\lambda^2 .\end{aligned}$$

with $\widetilde{U}^\top W(t_0) \widetilde{V} = 0$ by the construction of the augmented matrix $\widehat{S}$ used in (I), and in (II), we use Assumption 1.

Step 3 - Bounding of $\left\| \widehat{W}^r - W_1^r \right\|$: By construction of the truncation step we directly obtain

$$\left\| \widehat{W}^r - W_1^r \right\| \leq \vartheta$$

In conclusion, we obtain for a single iteration of Algorithm 1

$$\begin{aligned}\left\| W(t_1) - W_1^r \right\| \leq{}& \left\| W(t_1) - P_{\widehat{U}} W(t_1) P_{\widehat{V}} \right\| + \\ & \left\| \widehat{U}\widehat{U}^\top W(t_1) P_{\widehat{V}} - \widehat{W}^r \right\| + \left\| \widehat{W}^r - W_1^r \right\| \\ \leq{}& \widetilde{c}_1 \lambda\epsilon + \widetilde{c}_2 \lambda^2 + \vartheta\end{aligned}$$

To conclude, the global error in the training epochs follows by using the Lipschitz continuity of the gradient flow: We move from the local error in time to the global error in time by a standard ODEs argument of Lady Windermere's fan (Wanner & Hairer, 1996, §II.3); With $t = k\lambda$ and denoting the adapter computed with GeoLoRA at iteration k as W_t^r we then have

$$\|W(t) - W_t^r\| \leq c_1\epsilon + c_2\lambda + c_3\vartheta/\lambda\,.$$

This bounds the distance between the full-rank flow and GeoLoRA. The result trivially extends to the Riemannian flow of equation 4. Denote by $W_{\mathrm{RF}}(t)$ the solution of the Riemannian flow $\dot{W}_{\mathrm{RF}}(t) = -P(W_{\mathrm{RF}}(t))\nabla_W \mathcal{L}(W_{\mathrm{RF}}(t))$. Then, since $\|W(t) - W_{\mathrm{RF}}(t)\| \leq c\epsilon$, it directly follows that

$$\|W_{\mathrm{RF}}(t) - W_t^r\| \leq c_4\epsilon + c_2\lambda + c_3\vartheta/\lambda\,.$$

□

H VISUALIZATION OF THE STIFFNESS OF THE BASIC LOW-RANK SYSTEM

Consider Equation (5) in the case for $n = 20$. We set the target matrix

$$W = \begin{bmatrix} 0 & 15 & 0 & \dots \\ -2 & 0 & 0 & \dots \\ 0 & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \in \mathbb{R}^{20\times 20},$$

which has rank $r = 2$ and singular values $\sigma_1 = 15$ and $\sigma_2 = 2$ We compare SVD-lora, AdaLora, and GeoLoRA, both with an ansatz of form $W_{\text{ans}} = USV^\top$ initialized as

$$U, V = \begin{bmatrix} I \\ 0 \end{bmatrix} \in \mathbb{R}^{20\times 4}, \qquad S = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 1e-2 & 0 & 0 \\ 0 & 0 & 1e-4 & 0 \\ 0 & 0 & 0 & 1e-6 \end{bmatrix} \in \mathbb{R}^{4\times 4}$$

where U,V are orthonormal, and the S matrix has a fast decaying singular spectrum.

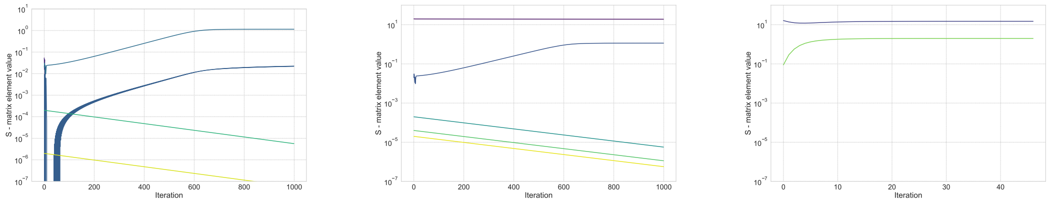
AdaLora and GeoLoRA use a relative singular value truncation threshold $\tau = 0.15$ for rank truncation. We found that learning rate $\lambda = 0.178$ is the maximal learning rate before AdaLora and SVD-Lora become unstable, whereas GeoLoRA allows for arbitrary large learning rates, and we set $\lambda = 0.1$. We present the trajectories of the S-matrix elements of the corresponding methods in Figure 5 for up to 1000 iterations or until single precision accuracy is reached. As seen in Figure 5, AdaLora and SVD-Lora exhibit heavy oscillations in the trajectories of the S-matrix elements - leading to slow convergence. Adalora - although using orthonormalization by regularization of the low-rank basis is not able to stabilize the training, leading to overestimation of the rank, which is $r = 5$ at final time and a final loss value of 1.6. Similarly SVD-Lora exhibits even stronger oscillations and is not able to find the right matrix approximation. In contrast, GeoLoRA identifies the correct rank $r = 2$ and the corresponding correct singular values 15 and 2.

I STRUCTURE PRESERVATION

The goal of this section is to clarify the formulation of Algorithm 1 in relation with the previous related literature. In particular, we want to show that the proposed algorithm can be seen as an efficient structure preservation formulation of a projected gradient flow (Koch & Lubich, 2007a) for training neural networks. In this section, to achieve full generality, we will denote with Y_i either the pretrained matrices or the low-rank adapters.

As already mentioned in the previous section, gradient descent can be seen an forward Euler discretization of the gradient system

$$\dot{Y}_i = -\nabla_{Y_i}\mathcal{L}(Y_1, \dots, Y_L), \;\; i = 1, \dots, L$$



(a) SVD-Lora, S matrix

(b) AdaLora, S matrix

(c) GeoLoRA, S matrix

Figure 5: Time-trace of the matrix elements of SVD-Lora (a) AdaLora (b) and the proposed method GeoLoRA (c) to solve Equation (5). SVD-Lora was trained with learning rate $\lambda = 0.00178$, which is the largest learning rate for which the optimization remained stable, GeoLoRA allows larger learning rates, set to $\lambda = 0.1$. GeoLoRA converges fast to single precision accuracy, whereas SVD-LORA still has a loss value of 1.7 after 1000 iterations, due to the heavy oscillations in it's S matrix trajectory (a). Adalora reduces the oscillations, however incorrectly identifies the rank and fails to converge due to the influence of the additional singular values.

The neural network $f_{Y_1,\dots,Y_L}$ naturally induces a weighted graph, where nodes are neurons and weights are connections among them and for which the adjacency matrix can be written as:

$$\mathcal{Y} := \left[\begin{array}{c|c|c|c|c} 0 & Y_1 & 0 & \cdots & 0 \\ \hline 0 & 0 & Y_2 & \cdots & 0 \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & 0 & \cdots & Y_L \\ \hline 0 & 0 & 0 & \cdots & 0 \end{array}\right] \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{N}|}$$

where $|\mathcal{N}| = \sum_{i=1}^{L} d_i$ is the total number of neurons of the neural network.
The matrix $\mathcal{Y}$ now represents the adjacency matrix of the computational graph, and the block structure is given by the layers. A model with a general full adjacency matrix $\mathcal{Y}$, would in general have non-zero connections between two generic layers i, j, descriebed by the block Y_{ij}. Let's consider the model

$$f_{\mathcal{Y}}(x) = z^L(x), \quad z^0(x) = x, \quad z^{\ell+1} = \sigma_i\Big(\sum_i Y_{i,\ell+1} z^i\Big)$$

Notice that for $\mathcal{Y}$ upper diagonal, the previous model would be a feedforward network. Given this observation, under the assumption that $f_{\mathcal{Y}}(x)$ is well defined as an eventual fixed point, we can now see the loss function as a function of the full adjacency matrix, with an abuse of notation we will call it again $\mathcal{L}(\mathcal{Y})$. Usual training would superimpose the sparse graph with the same structure of $\mathcal{Y}$, but let's consider for a moment the gradient flow

$$\dot{\mathcal{Y}} = -\nabla\mathcal{L}(\mathcal{Y})$$

Clearly, the flow does not preserve the sparsity of the adjacency matrix $\mathcal{Y}$, even for sparse initial conditions. Using the theory developed in (Koch & Lubich, 2007a) directly on this gradient flow would lead to neural networks with a non-feedforward topology. Moreover, given the size of $|\mathcal{N}|$ for modern neural networks, it can be expensive to compute QR or SVD decomposition of the basis matrices. Luckily, the sparsity structure is a simple linear constraint represented by the mask matrix

$$\mathcal{M} := \left[\begin{array}{c|c|c|c|c} 0 & 11^\top & 0 & \cdots & 0 \\ \hline 0 & 0 & 11^\top & \cdots & 0 \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & 0 & \cdots & 11^\top \\ \hline 0 & 0 & 0 & \cdots & 0 \end{array}\right] \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{N}|}$$

and the linear operator $\Pi(A) = \mathcal{M} \odot A$.

A system preserving the sparsity pattern is given naturally by the ODE

$$\dot{\mathcal{Y}} = -\Pi\nabla\mathcal{L}(\mathcal{Y})$$

However, it is not obvious that by projecting this last system on the manifold of rank-r matrices $\mathcal{M}_r$ the block structure is preserved. Fortunately, it is indeed the case, described by the following lemma:

Proposition 2. *(Block structure preservation of the flow)*
Consider the gradient flow with sparse initial condition

$$\dot{\mathcal{Y}} = -P(\mathcal{Y})\Pi\nabla\mathcal{L}(\mathcal{Y}), \;\; \mathcal{Y}(0) = \mathcal{Y}_0 \in range(\Pi)$$

Then $\mathcal{Y}(t) \in range(\Pi)$ *for all* $t \geq 0$*.*

Proof. It is necessary and sufficient to prove that $P(\mathcal{Y}(t))\Pi\nabla\mathcal{L}(\mathcal{Y}(t)) \in range(\Pi)$ for all $t \geq 0$, i.e. that $\Pi P(\mathcal{Y}(t))\Pi\nabla\mathcal{L}(\mathcal{Y}(t)) = P(\mathcal{Y}(t))\Pi\nabla\mathcal{L}(\mathcal{Y}(t))$. The key to prove this is to observe that for $Z \in range(\Pi)$, we have $P(\mathcal{Y})Z \in range(\Pi)$. In fact, given $Z \in range(\Pi)$, we can write a SVD of Z as

$$Z = \left[\begin{array}{c|c|c|c|c} 0 & U_1 & 0 & \cdots & 0 \\ \hline 0 & 0 & U_2 & \cdots & 0 \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & 0 & 0 & U_L \\ \hline I & 0 & 0 & \cdots & 0 \end{array}\right] \left[\begin{array}{c|c|c|c|c} 0 & 0 & 0 & \cdots & 0 \\ \hline 0 & S_1 & 0 & \cdots & 0 \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & 0 & S_{L-1} & 0 \\ \hline 0 & 0 & 0 & \cdots & S_L \end{array}\right] \left[\begin{array}{c|c|c|c|c} I & 0 & 0 & \cdots & 0 \\ \hline 0 & V_1^\top & 0 & \cdots & 0 \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & 0 & V_{L-1}^\top & 0 \\ \hline 0 & 0 & 0 & \cdots & V_L^\top \end{array}\right]$$

□

and we have $UU^\top, VV^\top \in range(\Pi)$. Thus, by direct calculation we can show that $UU^\top Z, ZVV^\top, UU^\top ZVV^\top \in range(\Pi)$ and thus $P(\mathcal{Y})Z = UU^\top Z + ZVV^\top - UU^\top ZVV^\top \in range(\Pi)$. Since $\Pi\nabla\mathcal{L}(\mathcal{Y}(t)) \in range(\Pi)$ by construction for all $t \geq 0$, we get the desider result. Thanks to this last proposition, following again the line of work in (Koch & Lubich, 2007a), it is possible to restrict the parameterization in the tangent space to a block-structured one as in Proposition 1. In this way, we get the following coherence theorem:

Proposition 3. *Consider the gradient flow with sparse initial condition*

$$U = \dot{\mathcal{Y}} = -P(\mathcal{Y})\Pi\nabla\mathcal{L}(\mathcal{Y}), \;\; \mathcal{Y}(0) = \mathcal{Y}_0 \in range(\Pi)$$

Consider now the parametrization $\mathcal{Y} = USV^\top$ *with*

$$U = \left[\begin{array}{c|c|c|c|c} 0 & U_1 & 0 & \cdots & 0 \\ \hline 0 & 0 & U_2 & \cdots & 0 \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & 0 & 0 & U_L \\ \hline I & 0 & 0 & \cdots & 0 \end{array}\right], S = \left[\begin{array}{c|c|c|c|c} 0 & 0 & 0 & \cdots & 0 \\ \hline 0 & S_1 & 0 & \cdots & 0 \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & 0 & S_{L-1} & 0 \\ \hline 0 & 0 & 0 & \cdots & S_L \end{array}\right], V = \left[\begin{array}{c|c|c|c|c} I & 0 & 0 & \cdots & 0 \\ \hline 0 & V_1 & 0 & \cdots & 0 \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & 0 & V_{L-1} & 0 \\ \hline 0 & 0 & 0 & \cdots & V_L \end{array}\right]$$

where $U_i^\top U_i = I, V_i^\top V_i = I$*. Then, by imposing the Gauge conditions* $\dot{U_i}^\top U_i = 0, \dot{V_i}^\top V_i = 0$*, the projected flow* $\dot{\mathcal{Y}} = -P(\mathcal{Y})\Pi\nabla\mathcal{L}(\mathcal{Y})$ *can be rewritten in block fashion as follows:*

$$\begin{aligned} \dot{S}_i(t) &= -U_i^\top(t)\nabla_{Y_i}\mathcal{L}(U(t)S(t)V(t)^\top)V_i(t), \\ \dot{U}_i(t) &= -\left(I - P_{U_i(t)}\right)\nabla_{Y_i}\mathcal{L}(U(t)S(t)V(t)^\top)V_i(t)S_i(t)^{-1}, \\ \dot{V}_i(t) &= -\left(I - P_{V_i(t)}\right)\nabla_{Y_i}\mathcal{L}(U(t)S(t)V(t)^\top)U_i(t)S_i(t)^{-\top}, \quad i = 1, \ldots, L \end{aligned}$$

Proof. Thanks to the previous proposition, we know that the variation $P(\mathcal{Y})\Pi\nabla\mathcal{L}(\mathcal{Y}) \in range(\Pi)$ for all $t \geq 0$. Then, we have $\mathcal{Y}(t) \in range(\Pi)$ for all times, and thus we can decompose it using a block SVD as described in the statement of the proposition. Moreover, by the self-adjointness of Π, Galerkin condition can be written as:

$$\langle \dot{\mathcal{Y}} + \nabla\mathcal{L}(\mathcal{Y}), q\rangle = \langle \dot{U}SV^\top + U\dot{S}V^\top + US\dot{V}^\top + \nabla\mathcal{L}(\mathcal{Y}), q\rangle = 0, \;\; \forall q \in T_{\mathcal{Y}}\mathcal{M}_r \cap range(\Pi)$$

Since $q \in T_{\mathcal{Y}}\mathcal{M}_r \cap range(\Pi)$, we can represent it as $q = \delta USV^\top + U\delta SV^\top + US\delta V^\top$, with $\delta U, \delta V, \delta S$ with the same block structure of U, S and V. By writing the last conditions on a basis of $T_{\mathcal{Y}}\mathcal{M}_r \cap range(\Pi)$, we get

$$\begin{aligned} \langle \dot{U}SV^\top + U\dot{S}V^\top + US\dot{V}^\top + \nabla\mathcal{L}(\mathcal{Y}), \delta USV^\top\rangle = 0 \\ \langle \dot{U}SV^\top + U\dot{S}V^\top + US\dot{V}^\top + \nabla\mathcal{L}(\mathcal{Y}), U\delta SV^\top\rangle = 0 \\ \langle \dot{U}SV^\top + U\dot{S}V^\top + US\dot{V}^\top + \nabla\mathcal{L}(\mathcal{Y}), US\delta V^\top\rangle = 0 \end{aligned}$$

Thanks to the Gauge conditions $\dot{U}^\top U = 0, \dot{V}^\top V = 0$ and to the properties of the Frobenius inner product, the last system becomes

$$\begin{aligned}
&\langle \dot{U} S S^\top + \nabla\mathcal{L}(\mathcal{Y}) V S^\top, \delta U\rangle = 0 \\
&\langle U^\top U \dot{S} V^\top V + U^\top \nabla\mathcal{L}(\mathcal{Y}) V, \delta S\rangle = \langle \dot{S} + U^\top \nabla\mathcal{L}(\mathcal{Y}) V, \delta S\rangle = 0 \\
&\langle S^\top S \dot{V}^\top + S^\top U^\top \nabla\mathcal{L}(\mathcal{Y}), \delta V^\top\rangle = 0
\end{aligned}$$

and from this equations we get the known

$$\begin{aligned}
\dot{U} &= -(I - UU^\top)\nabla\mathcal{L}(\mathcal{Y}) V S^{-1} \\
\dot{S} &= -U^\top \nabla\mathcal{L}(\mathcal{Y}) V \\
\dot{V} &= -(I - VV^\top)\nabla\mathcal{L}(\mathcal{Y})^\top U S^{-\top}
\end{aligned}$$

By writing this equations block-by-block, we get the desidered result. □

This last proposition clarifies how to connect the single matrix setting with the multi-matrix setting, showing that the presentation of Algorithm 1 is in fact coherent with the single matrix setting. Moreover, investigation of this setting leads naturally to the global truncation strategy.

I.1 TRUNCATION STRATEGY

The global truncation strategy proposed in the main manuscript is in fact coherent with the single matrix formulation presented in the previous section. In fact, one can assemble the rank augmented S matrix as:

$$\widehat{S}(t=1) = \left[\begin{array}{c|c|c|c|c}
\widehat{S}_1 & 0 & 0 & \cdots & 0 \\ \hline
0 & \widehat{S}_2 & 0 & \cdots & 0 \\ \hline
\vdots & \vdots & \vdots & \ddots & \vdots \\ \hline
0 & 0 & 0 & \widehat{S}_{L-1} & 0 \\ \hline
0 & 0 & 0 & \cdots & \widehat{S}_L
\end{array}\right]$$

and then truncate the smallest singular values up to required precision. This can be efficiently done by computing an SVD on each diagonal block, giving effectively an SVD of the global matrix. In particular, if $\widehat{S}_i = P_i \Sigma_i Q_i^\top$ we get that

$$\widehat{S} = blockdiag(P_1, \dots, P_L) blockdiag(\Sigma_1, \dots, \Sigma_L) blockdiag(Q_1, \dots, Q_L)^\top$$

Since the matrix $blockdiag(\Sigma_1, \dots, \Sigma_L)$ is effectively diagonal, by assuming the diagonal is increasingly ordered, it is natural to globally truncate the ranks according to the minimal k such that

$$\frac{\sum_{i=k+1}^{2rL} \sigma_i^2}{\sum_{i=1}^{2rL} \sigma_i^2} < \frac{\tau}{1-\tau}$$

Which corresponds in throwing away the smallest singular values of $\widehat{S}$ until we reach the desired relative error. By rewriting this criterion on the singular values of each matrix $\widehat{S}_i$, we get exactly the global criterion proposed in Section 4.

J OPTIMALITY ON THE LOW-RANK MANIFOLD

We remark below that if $W_\star$ is a local minimum, then $P(W_\star)\nabla\mathcal{L}(W_\star) = 0$. In particular, since $P(W)\nabla\mathcal{L}(W)$ is the Riemannian gradient with respect to the ambient metric, then the following holds by definition of the gradient:

$$\partial_{\delta W}\mathcal{L}(W) = \langle P(W)\nabla\mathcal{L}(W), \delta W\rangle$$

where $\partial_{\delta W}\mathcal{L}(W)$ is the directional derivative of $\mathcal{L}$ along the direction δW. Thus, $P(W_\star)\nabla\mathcal{L}(W_\star) = 0$ if and only if $\partial_{\delta W}\mathcal{L}(W) = 0$ for all $\delta W \in T_W\mathcal{M}$ and this happens if and only if $\nabla\mathcal{L}(W) \in (T_W\mathcal{M})^\perp$. So geometrically, if $W_\star$ is a local minimum, then $P(W_\star)\nabla\mathcal{L}(W_\star) = 0$ means that among all available directions, there are none that decrease the loss.

For simultaneous descent, the same condition doesn't hold, in fact, the algorithm's stationary points satisfy $\widehat{P}(W_\star)\nabla\mathcal{L}(W_\star) = 0$, which given the non-orthogonality does not, in general, imply $P(W_\star)\nabla\mathcal{L}(W_\star) = 0$, so there could be descent directions unexploited by the method.