Riemannian Preconditioned LoRA for Fine-Tuning Foundation Models

Fangzhao Zhang [1] **Mert Pilanci** [1]

Abstract

Low-Rank Adaptation (LoRA) emerges as a popular parameter-efficient fine-tuning (PEFT) method, which proposes to freeze pretrained model weights and update an additive low-rank trainable matrix. In this work, we study the enhancement of LoRA training by introducing an $r \times r$ preconditioner in each gradient step where r is the LoRA rank. We theoretically verify that the proposed preconditioner stabilizes feature learning with LoRA under infinite-width NN setting. Empirically, the implementation of this new preconditioner requires a small change to existing optimizer code and creates virtually minuscule storage and runtime overhead. Our experimental results with both large language models and text-to-image diffusion models show that with this new preconditioner, the convergence and reliability of SGD and AdamW can be significantly enhanced. Moreover, the training process becomes much more robust to hyperparameter choices such as learning rate. The new preconditioner can be derived from a novel Riemannian metric in low-rank matrix field. Code can be accessed at https://github.com/pilancilab/Riemannian_Preconditioned_LoRA.

1. Introduction

With the expanding scale of neural network models in both vision and language domains, training a neural network from scratch to match the performance of existing large models has become almost infeasible. As a result, fine-tuning has emerged as a prevalent approach for downstream tasks. Traditional full parameter fine-tuning demands extensive storage, making it impractical for many applications. In contrast, recent advances in Parameter-Efficient Fine-Tuning (PEFT) methods offer a more storage-efficient solution while still delivering strong performance in downstream tasks. A widely-used PEFT method is Low Rank Adaptation, also known as LoRA (Hu et al., 2021), which proposes to add low-rank matrices to existing model weights and only train these additive components. Simply speaking, for a pretrained model weight matrix W of dimension m by n, LoRA replaces W with $W + BA$ where B, A are trainable weight matrices of dimension m by r and r by n for some small rank r. In the fine-tuning procedure, W is frozen and we are only optimizing over A's and B's. Compared to full fine-tuning, LoRA introduces fewer trainable parameters. Effectiveness of LoRA has been empirically verified in different fields. Here we note that optimizing LoRA parameters falls into optimizing over low-rank matrices which form a quotient manifold. This motivates the idea of enhancing LoRA training via tools from the field of Riemannian optimization.

Recent work such as LoRA+ (Hayou et al., 2024) has drawn attention to the optimization paradigm of LoRA and reveals that the learning rate of LoRA parameter B should be set larger than that of A to achieve stable feature learning under the infinite-width NN setting, making the learning rate tuning a joint hyperparameter search. More specifically, denote learning rates for A and B by η_A and η_B respectively, LoRA+ proposes to use a heuristic $\eta_B/\eta_A = 2^4$ in practice and tune only η_A. In this work, we propose an improvement of LoRA training with the introduction of an $r \times r$ preconditioner in its optimization step, and we show that with this simple preconditioner, LoRA learns stable features without the need of setting different learning rates for A and B. As an illustration, we propose modifying the gradient updates for the LoRA parameters as follows

$$\begin{aligned} A_{t+1} &= A_t - \alpha(B_t^T B_t)^{-1}(\nabla_{A_t}\mathcal{L}), \\ B_{t+1} &= B_t - \alpha(\nabla_{B_t}\mathcal{L})(A_t A_t^T)^{-1}, \end{aligned} \tag{1}$$

where $(A_t A_t^T)^{-1}$ and $(B_t^T B_t)^{-1}$ are the preconditioners we introduce and $\mathcal{L}$ is the training loss objective. This scaled gradient method (scaled GD) can be derived from a novel Riemannian metric studied in (Mishra et al., 2012) which takes into consideration both the objective function and constraints and has been shown to have better convergence rate for conventional low-rank matrix optimization problems involving matrix sensing and robust PCA (Candes et al., 2009). Our work shows for the first time that

[1]Department of Electrical Engineering, Stanford University. Correspondence to: Fangzhao Zhang <zfzhao@stanford.edu>.

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

Figure 1. Generation results for prompt "a blue $\langle V_{\text{vase}} \rangle$" after fine-tuning on 6 red vase images of the Stable Diffusion V1.5 model. No black images are observed for our method (scaled AdamW)'s generation and AdamW generates only black images for large learning rates. Our method generates photos better capturing the prompt and is more robust to learning rate changes. See Section 6.4.1 for experimental details.

this preconditioner enables LoRA to achieve stable feature learning with Adam optimizer, revealing the superiority of preconditioning in fine-tuning deep learning models. Empirically, we verify the validity of this new preconditioner for LoRA training via extensive fine-tuning tasks for both language models and text-to-image diffusion models. The experimental results show that the convergence of both SGD and AdamW is significantly enhanced by preconditioning (gradient scaling). Despite the accelerated convergence, the optimization procedure also becomes more robust to learning rate changes with this new preconditioner. Figure 1 compares the generation results for diffusion model fine-tuning with AdamW optimizer with and without gradient scaling, from which it can be observed that gradient scaling significantly improves the generation quality as well as training robustness against learning rate changes, see Algorithm 1 for the formal algorithm for scaled AdamW. Most importantly, our preconditioner is only of dimension r by r thus the storage overhead is negligible for small LoRA rank r. Also, unlike most second-order preconditioners based on Hessian inverses, inverting an r by r matrix for small r introduces negligible runtime overhead and runs as fast as unpreconditioned optimizers. In practice, small values, e.g., $r = 4$, are usually used as default for LoRA fine-tuning, see Figure 2 for runtime comparison when fine-tuning GPT-2 with both scaled and unscaled optimizers.

To summarize, we study the application of preconditioned gradient updates (1) in LoRA training. Theoretically, we show that the proposed preconditioner provides an elegant solution to learning rate choices for stable feature learning with LoRA under infinite-width NN setting. We also provide a convergence guarantee for fine-tuning a reparameterized model which is equivalent to a two-layer ReLU network via convexification. See Section 4 and Section 7 respectively for details. Empirically, we apply this method for LoRA fine-tuning for both large language and diffusion models and observe that scaled optimizers significantly improve the performance of unscaled optimizers. See Section 6 for the experimental details. To the best of our knowledge, this is the first work to apply Riemannian optimization in designing preconditioners for fine-tuning large foundation models. This approach is particularly appropriate given the matrix factorization characteristics of the LoRA model.

Next, we first provide some intuition on our method in Section 3. We then show that LoRA training with the proposed preconditioners achieves stable feature learning in Section 4. We include an overview of basic Riemannian optimization concepts as well as the derivation of this preconditioner by introduction of a new Riemannian metric in Section 5. We offer a pseudocode and present extensive experimental results in Section 6. Finally we state our convergence results for related problems in Section 7 and review prior literature in Section 8.

2. Notation

We adopt same notation convention as in LoRA+ (Hayou et al., 2024). Specifically, for any sequence $s_n \in \mathbb{R}$ and any given $c_n \in \mathbb{R}^+$, we use $s_n = \mathcal{O}(c_n)$ and $s_n = \Omega(c_n)$ to represent $s_n < \kappa c_n$ and $s_n > \kappa c_n$ respectively for some $\kappa > 0$. We use $s_n = \Theta(c_n)$ when both $s_n = \mathcal{O}(c_n)$ and $s_n = \Omega(c_n)$. For vector and matrix sequences, the notation is applied entrywise. For a sequence that is a vector of random variables, convergence in second moment, i.e., L_2 convergence, is adopted.

3. Theoretical Insights

We first offer an intuitive explanation for the effectiveness of the preconditioner (1) before we move on to its stable learning properties and review its rigorous derivation from a Riemannian metric formulation. Consider in the t-th iteration pretrained model weight W and its additive low-rank component B_tA_t. Let $X_t = W + B_tA_t$ denote the

whole weight matrix and let $\mathcal{L}$ denote the loss function, i.e., $\mathcal{L}(A, B) := \mathcal{L}(W + BA)$. For the plain gradient descent method, when step size η is small, the updated weight is approximately given by

$$\begin{aligned} X_{t+1} &= W + B_{t+1}A_{t+1} \\ &\approx W + B_tA_t - \eta\nabla_{B_t}\mathcal{L}A_t - \eta B_t\nabla_{A_t}\mathcal{L} \\ &= X_t - \eta\nabla_{X_t}\mathcal{L}(A_t^TA_t) - \eta(B_tB_t^T)\nabla_{X_t}\mathcal{L}, \end{aligned}$$

where we ignore the second-order term in the second line since when η is small, η^2 is negligible. The derivation from the second line to the third line comes from the simple fact $\nabla_{B_t}\mathcal{L} = (\nabla_{X_t}\mathcal{L})A_t^T$ and $\nabla_{A_t}\mathcal{L} = B_t^T(\nabla_{X_t}\mathcal{L})$. Thus the LoRA update in gradient descent step is approximately constrained to the column space of B_t and the row space of A_t. If we scale $\nabla_{B_t}\mathcal{L}$ right by $(A_tA_t^T)^{-1}$ and scale $\nabla_{A_t}\mathcal{L}$ left by $(B_t^TB_t)^{-1}$, which is exactly the preconditioner (1), the scaled update becomes

$$\begin{aligned} \tilde{X}_{t+1} \approx X_t &- \eta(\nabla_{X_t}\mathcal{L})A_t^T(A_tA_t^T)^{-1}A_t \\ &- \eta B_t(B_t^TB_t)^{-1}B_t^T(\nabla_{X_t}\mathcal{L}) \\ = X_t &- \eta\mathrm{Proj}_{\mathrm{row}(A_t)}(\nabla_{X_t}\mathcal{L}) - \eta\mathrm{Proj}_{\mathrm{col}(B_t)}(\nabla_{X_t}\mathcal{L})^T, \end{aligned}$$

where the update is done according to projection of the full matrix gradient onto the row space of A_t and the column space of B_t, which better approximates full fine-tuning compared to the unscaled gradient descent step. Therefore, our preconditioner (1) effectively serves as a gradient projector.

4. Stable Feature Learning

Given the current trend of increasing model sizes, there has been a focus on analyzing the asymptotic training behavior of neural networks as the number of neurons approaches infinity (Schoenholz et al., 2017; Hayou et al., 2019; Yang, 2020). Under this infinite-width NN setting, we naturally expect that both the NN prediction $f^t(x)$ in the t-th iteration and the increment $\Delta f^t := f^t(x) - f^{t-1}(x)$ to be of constant magnitude, which ensures that neither the NN predictions nor the increments explode or vanish as the NN size increases, thereby leading to stable training dynamics. We refer to this behavior as *stable feature learning*, formally defined in Definition A.1 in the Appendix. The authors of LoRA+ (Hayou et al., 2024) observe that the learning rate of LoRA parameter B should be set to be larger than that of A for asymptotic stable feature learning. Their analysis mostly focuses on the order of magnitude of iterates and the conclusion that $\eta_B \gg \eta_A$ does not immediately offer practical guidance. Here we show that our preconditioned updates (1) introduce an elegant and practical solution to learning rate choices for stable feature learning in the infinite-width NN regime. Specifically, we will see that LoRA trained with Adam optimizer scaled by our preconditioner as in (1) leads to stable feature learning when the same learning rate is used for training A and B. In contrast, LoRA trained with the unpreconditioned Adam optimizer requires different learning rate settings for A and B to achieve stable feature learning.

We first illustrate our key point via a simple toy example and we then proceed to our main theorem. Consider the simple linear model

$$f(x) = (W + ba^T)x,$$

where $W \in \mathbb{R}^{1\times n}$ is the pretrained model weight and $b \in \mathbb{R}, a \in \mathbb{R}^n$ are trainable LoRA parameters. Consider the quadratic loss function $\mathcal{L}(a, b) = (f(x) - y)^2/2$ with some scalar label y. We adopt Gaussian initialization $a_i \sim \mathcal{N}(0, \sigma_a^2), b \sim \mathcal{N}(0, \sigma_b^2)$. Conventionally, ba^T is initialized at zero for LoRA, and we thus consider setting $\sigma_a^2 = 0, \sigma_b^2 = \Theta(1)$.

Analysis of unpreconditioned training. Assume the model is trained with gradient descent with learning rate $\eta = \Theta(n^c)$ for some $c \in \mathbb{R}$. Since the training procedure involves only elementary algebraic operations, the quantities there should be of powers of n. In iteration t, the feature update without preconditioning is given by

$$\begin{aligned} \Delta f_t &:= f_t(x) - f_{t-1}(x) \\ &= -\eta b_{t-1}^2(f_{t-1}(x) - y)\|x\|^2 \\ &\quad - \eta(a_{t-1}^Tx)^2(f_{t-1}(x) - y) \\ &\quad + \eta^2(f_{t-1}(x) - y)^2b_{t-1}(a_{t-1}^Tx)\|x\|^2. \end{aligned}$$

We denote $\delta_t^1 = \eta b_{t-1}^2(f_{t-1}(x) - y)\|x\|^2, \delta_t^2 = \eta(a_{t-1}^Tx)^2(f_{t-1}(x) - y)$, and $\delta_t^3 = \eta^2(f_{t-1}(x) - y)^2b_{t-1}(a_{t-1}^Tx)\|x\|^2$. For stable feature learning, we would like $\delta_t^1, \delta_t^2, \delta_t^3 \in \Theta(1)$ and further $f_{t-1}(x) \in \Theta(1)$. Note that $\delta_t^3 \in \Theta(1)$ is guaranteed as long as $\delta_t^1, \delta_t^2 \in \Theta(1)$. Thus for stable feature learning, it suffices to have $\delta_t^1, \delta_t^2, f_{t-1}(x) \in \Theta(1)$. For the sake of notational clarity, we introduce new notation γ such that $v = \Theta(n^{\gamma[v]})$ captures the polynomial behavior for any v. We refer readers to Section A.2 in (Hayou et al., 2024) for additional properties of $\gamma[\cdot]$. Stable feature learning is thus equal to the following linear constraints

$$\begin{cases} c + 2\gamma[b_{t-1}] + 1 = 0 \quad (\text{for } \delta_t^1 = \Theta(1)), \\ c + 2\gamma[a_{t-1}^Tx] = 0 \quad (\text{for } \delta_t^2 = \Theta(1)), \\ \gamma[b_{t-1}] + \gamma[a_{t-1}^Tx] = 0 \quad (\text{for } f_{t-1}(x) = \Theta(1)), \end{cases}$$

from which we can derive $c = -1/2$ and thus the learning rate should be set to $\eta = \Theta(n^{-1/2})$. With $\gamma[b_1] = \gamma[b_0] = 0$ and $\gamma[a_1^Tx] = \gamma[\eta b_0 y\|x\|^2] = 1/2$, one can inductively deduce that $\gamma[b_t] = 0$ and $\gamma[a_t^Tx] = 1/2$ for all t and thus $\gamma[f_t] = 1/2$, contradicting to $f_t = \Theta(1)$. Instead, stable feature learning requires to set separately $\eta_a = \Theta(1/n)$ and $\eta_b = \Theta(1)$ as shown in Proposition 2 in (Hayou et al., 2024).

Analysis of preconditioned training. We now proceed to show that with our preconditoned parameter updates (1), $c = -1$ would generate stable feature learning. After the injection of preconditioner (1), the training dynamics are given by

$$\begin{aligned}\Delta f_t &:= f_t(x) - f_{t-1}(x)\\ &= (b_{t-1} - \eta(f_{t-1}(x) - y)a_{t-1}^T x \underbrace{\|a_{t-1}\|^{-2}}_{\text{preconditioner}})(a_{t-1}^T\\ &\quad - \eta(f_{t-1}(x) - y)b_{t-1}x^T \underbrace{b_{t-1}^{-2}}_{\text{preconditioner}})x\\ &= -\eta(f_{t-1}(x) - y)\|x\|^2\\ &\quad - \eta(a_{t-1}^T x)^2(f_{t-1}(x) - y)\|a_{t-1}\|^{-2}\\ &\quad + \eta^2(f_{t-1}(x) - y)^2 b_{t-1}^{-1}\|a_{t-1}\|^{-2}(a_{t-1}^T x)\|x\|^2.\end{aligned}$$

We similarly define $\delta_t^{1\text{scaled}} = \eta(f_{t-1}(x) - y)\|x\|^2$, $\delta_t^{2\text{scaled}} = \eta(a_{t-1}^T x)^2(f_{t-1}(x) - y)\|a_{t-1}\|^{-2}$, $\delta_t^{3\text{scaled}} = \eta^2(f_{t-1}(x) - y)^2 b_{t-1}^{-1}\|a_{t-1}\|^{-2}(a_{t-1}^T x)\|x\|^2$ as scaled version of $\delta_t^1, \delta_t^2, \delta_t^3$. For stable feature learning, we thus have the below modified linear constraints

$$\begin{cases} c + 1 = 0 \quad (\text{for } \delta_t^{1\text{scaled}} = \Theta(1)),\\ c + 2\gamma[a_{t-1}^T x] - \gamma[\|a_{t-1}\|^2] = 0 \quad (\text{for } \delta_t^{1\text{scaled}} = \Theta(1)),\\ \gamma[b_{t-1}] + \gamma[a_{t-1}^T x] = 0 \quad (\text{for } f_{t-1}(x) = \Theta(1)),\end{cases}$$

from which we can derive $c = -1$. With $\eta = \Theta(n^{-1})$, we get $\gamma[b_1] = \gamma[b_0] = 0$ and $\gamma[a_1^T x] = \gamma[\eta b_0^{-1} y\|x\|^2] = 0$. One can recursively derive $b_t, a_t^T x, \delta_1^{\text{scaled}}, \delta_2^{\text{scaled}}, \delta_3^{\text{scaled}} \in \Theta(1)$ for all t, which preserves $f_t \in \Theta(1)$ and $\Delta f_t \in \Theta(1)$, i.e., stability is achieved.

The above toy example illustrates that our preconditioned LoRA updates achieve stable feature learning with learning rates for A and B of same order of magnitude, while for unpreconditioned LoRA, different learning rates are required to obtain the same stability. Nevertheless, the toy example is limited to linear model with LoRA rank $r = 1$ and also gradient updates without a momentum term. Indeed, the stability persists for arbitrary LoRA ranks and for the Adam optimizer when aided with our preconditioner (1), which we formalize as our theorem below.

Theorem 4.1. *[Stable Feature Learning (Informal)] Assume LoRA parameters A and B are trained with Adam scaled by our preconditioner as in (1). Further assume that BAx has dimension $\Theta(n)$. Then the LoRA model achieves stable feature learning with $\eta = \Theta(1)$. While for unscaled Adam, $\eta_A = \Theta(n^{-1})$ and $\eta_B = \Theta(1)$ are required for stable feature learning.*

Note here we require $\eta = \Theta(1)$ instead of $\eta = \Theta(n^{-1})$ as in the toy example, this discrepancy is due to our assumption about vector output and Adam optimizer's gradient processing. See Section A for the detailed statement and the proof of Theorem 4.1 as well as an explanation of this difference. Theorem 1 in (Hayou et al., 2024) shows that $\eta_A = \Theta(n^{-1})$ and $\eta_B = \Theta(1)$ are required for LoRA training with unpreconditioned Adam to obtain stability, revealing that without preconditioning we need to tune η_A and η_B separately while our preconditioner elegantly fixes this imbalance between the learning rates.

5. A Riemannian Metric Formulation

After discussing the motivation and superiority for stabilizing feature learning of our proposed preconditioner in prior sections, we now review how the proposed preconditioner is derived from a new Riemannian metric. Optimization over matrix with rank constraint is a common example for optimization over Riemannian submanifolds. Specifically, matrices with fixed rank form a quotient manifold of general matrix field. Let $\mathcal{M}$ denote any Riemannian submanifold, then Riemannian gradient descent usually takes form

$$X_{t+1} = \mathcal{R}(X_t - \eta\nabla_{M_r} f(X_t)),$$

where $\mathcal{R}$ is a retraction operator that maps to $\mathcal{M}$. Here, f is the objective function and $\nabla_{M_r} f(X_t)$ denotes Riemannian gradient defined by

$$Df(x)[\eta_x] = g_x(\nabla_{M_r} f(x), \eta_x) \text{ for all } \eta_x \in T_x\mathcal{M},$$

where $Df(x)[\eta_x]$ is the conventional Euclidean directional derivative of f in the direction η_x. In this definition, g_x is a Riemannian metric which maps two elements in the tangent space $T_x\mathcal{M}$ to a real number and might not be unique, we will see that the innovative design of g_x is the key to derive our preconditioned updates (1). Before proceeding to the derivation of our preconditioner, we need one more piece of knowledge about quotient manifold.

When it comes to a quotient space $\mathcal{M}/\sim$ where each element $[x] = \{y \in \mathcal{M} : y \sim x\}$ represents an equivalence class, for low-rank matrix problems where AB^T are considered, each (A, B) pair is equivalent to (AO, BO^{-1}) for any $O \in GL(r)$ in the sense that they obtain the same objective value, where $GL(r)$ stands for the general linear group over $r \times r$ invertible matrices. Tangent space $T_{[x]}\mathcal{M}$ respects the equivalence relation $\sim$ by the introduction of horizontal and vertical spaces at each element, i.e., we decompose $T_x\mathcal{M} = \mathcal{V}_x \oplus \mathcal{H}_x$ where $\mathcal{V}_x$ is the tangent space of the equivalence class $[x]$ and $\mathcal{H}_x$ is its complement. Then each $\eta_{[x]} \in T_{[x]}\mathcal{M}$ corresponds to a unique element in $\mathcal{H}_x$ which is called the horizontal lift of x. A Riemannian metric for $\mathcal{M}/\sim$ satisfies

$$g_{[x]}(\eta_{[x]}, \xi_{[x]}) = g_x(\eta_x, \xi_x),$$

where η_x and ξ_x are horizontal lifts of $\eta_{[x]}$ and $\xi_{[x]}$ at x. Thus a Riemannian metric on quotient space is invariant along equivalence classes of the quotient space.

Algorithm 1 Pseudocode of scaled AdamW in PyTorch.

```
# group trainable parameters into LoRA pairs in train.py.
for LoRA_A, LoRA_B in pairwise(trainable_parameter):
    param_groups.append({"params":  [LoRA_A,LoRA_B], "lr":  learning_rate})
# apply preconditioner in optimizer.py
for group in param_groups:
     A, B = group["params"]
     dA, dB = group["params"].grad
     # update parameter A
     dA_scaled =inverse(B.T@B+delta*torch.eye(r)).mm(dA)  # precondition gradient of A
     A_m = beta1*A_m+(1-beta1)*dA_scaled; A_m_hat = A_m/(1-beta1**t) # update first momentum of A
     A_v = beta2*A_v+(1-beta2)*dA_scaled**2; A_v_hat = A_v/(1-beta2**self.t) # update second momentum of A
     A.add_(A_m_hat/(sqrt(A_v_hat)+eps), -group['lr']) # update A
     # update parameter B similarly
     # ...
```

`pairwise`: read every two elements in a list

In (Mishra & Sepulchre, 2016), Mishra et al. describe a new Riemannian metric that draws motivation from regularized Lagrangian and involves both objectives and constraints. When specialized to least squares matrix decomposition problem of form $\|AB^T - Y\|_F^2/2$, following derivation (33) in (Mishra & Sepulchre, 2016), we get the following metric on quotient space $[x] = (A, B)$,

$$g_{[x]}(\eta_{[x]}, \xi_{[x]}) = \langle \eta_A, \xi_A B^T B \rangle + \langle \eta_B, \xi_B A^T A \rangle. \quad (2)$$

Under this new metric, the Riemannian gradient descent effectively replaces the gradient operators via the map

$$\left(\frac{\partial}{\partial A}\right) \to \left(\frac{\partial}{\partial A}\right)(B^T B)^{-1}, \left(\frac{\partial}{\partial B}\right) \to \left(\frac{\partial}{\partial B}\right)(A^T A)^{-1},$$

which then corresponds to the preconditioned updates (1) we propose in this work. For details in the design of the new metric (2) and its connection with sequential quadratic programming (SQP), we point readers to (Mishra & Sepulchre, 2016) and (Mishra et al., 2012) which include a thorough explanation and visualization of Riemannian optimization concepts.

6. Empirical Results

6.1. Algorithms and Simple Implementation

In this section, we describe the optimization algorithms and the software implementation we use for accelerating LoRA training in practice. Let $\mathcal{L}$ denote the loss function and $(A^{(t)}, B^{(t)})$ denote the pair of LoRA parameters in the t-th iteration.

Scaled GD. To apply gradient scaling to SGD, we follow exactly (1) and use $(B^{(t-1)^T} B^{(t-1)} + \delta I)^{-1}$ to scale gradient $\nabla_{A^{(t-1)}} \mathcal{L}$ and vice versa. Note here a small $\delta > 0$ is used to tackle the case when either $B^{(t-1)^T} B^{(t-1)}$ or $A^{(t-1)} A^{(t-1)^T}$ is not invertible. See Appendix C for the complete algorithm.

Scaled AdamW. The conventional scaled GD method studied for classic low-rank matrix optimization problems is only based on the gradient descent method. Our Theorem 4.1 introduces this preconditioner for Adam for the first time and reveals its advantages for Adam and its variants. We note that AdamW is more popular than SGD for fine-tuning due to its fast convergence. To extend preconditioning to AdamW, one could apply the preconditioner at each individual gradient computation step or apply the scaling to the processed gradient. Though our proof of Theorem 4.1 adopts the latter version and preconditions the processed gradient, we empirically find that scaling the gradient in each single iteration behaves better than scaling the processed gradient, thus we scale each single gradient in AdamW algorithm for our practical implementation and dub it scaled AdamW method. We outline the pseudocode of our scaled AdamW in Algorithm 1. Remarkably, our method only requires four lines change of existing optimizer code, which is simple to implement. We highlight the changed code in red color.

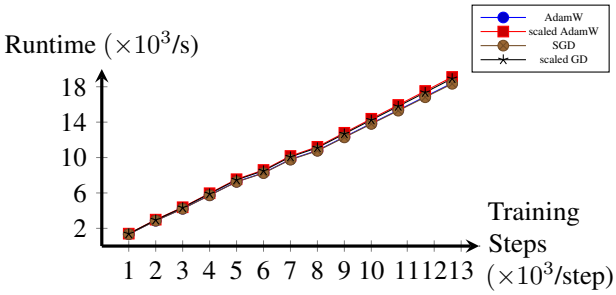

Figure 2. Runtime for LoRA fine-tuning GPT-2 medium model with different optimizers. Our scaled methods introduce negligible runtime overhead and train as fast as unscaled methods. See Section 6.2 for experimental details. Here we set $r = 4$.

6.2. Runtime Comparison

A main concern of our method may arise from the common stereotype about the cumbersomeness and heavy computation complexity of usual preconditioning methods. This is likely due to Hessian-inverse type second-order precondi-

Method	E2E				
	BLEU	NIST	MET	ROUGE-L	CIDEr
$\text{SGD}_{r=4}$	66.6	8.54	44.2	68.2	2.32
scaled GD (ours)$_{r=4}$	69.2	8.71	46.3	70.9	2.48
$\text{AdamW}_{r=4}$	68.9	8.69	46.5	71.3	2.51
scaled AdamW (ours)$_{r=4}$	**69.6**	**8.77**	**46.6**	**71.8**	**2.52**

Table 1. Scores for LoRA fine-tuning of GPT-2 medium model on E2E Natural Language Generation challenge with different optimizers. Our scaled optimizers outperform unscaled optimizers on all evaluation metrics and scaled GD closes the performance gap between SGD and AdamW. See Section 6.3.1 for experimental details.

tioner usually involves large size preconditioners and complex computation procedures, which is not the case for the preconditioners we consider. In each iteration, we use current value of $(AA^T)^{-1}$ to precondition gradient of B. The preconditioner is easily obtained and is of small size. We present a runtime comparison between scaled optimizers and their unscaled counterparts for fine-tuning a GPT-2 medium model on E2E NLG challenge, see Section 6.3.1 for experimental details. Figure 2 shows the runtime used for different optimizers for the fine-tuning task trained on NVIDIA A100 GPUs. Note there is little difference between the scaled optimizers and unscaled ones, which verifies that our preconditioner is practical. See runtime comprisons for larger rank $r = 256$ in Appendix D.

6.3. LLM Fine-Tuning

In this section, we study the fine-tuning task for GPT-2 model and Mistral 7B model with our scaled optimizers. Empirically, we observe that our scaled optimizers outperform unscaled optimizers by a large margin for various tasks, datasets, LoRA ranks, model sizes, model types, and benchmarks, which demonstrates the superiority of gradient scaling in training LoRA models. See below and also Appendix E for all our experiments.

6.3.1. GPT-2

We exploit the new preconditioner for LoRA fine-tuning of GPT-2 models. We follow exactly the same experimental setup as (Hu et al., 2021) except that here we tune learning rate individually using grid search for different methods being tested, see Appendix E.1 for experimental details and training hyperparameters. Table 1 shows the final score for fine-tuning GPT-2 medium model with LoRA rank 4 on E2E (Novikova et al., 2017) natural language generation challenge. It can be observed that scaled GD method closes the gap between SGD and AdamW and behaves comparable to AdamW while demanding smaller optimizer storage. Scaled AdamW method improves score of AdamW method on all evaluation metrics, which reveals that the new preconditioner is advantageous even for gradient computation normalized by gradient variance in AdamW method. See also Appendix E.1 for experimental results for different LoRA ranks, different datasets, and different model sizes. Our scaled optimizers show significant and uniform improvements over unscaled optimizers for almost all tests.

6.3.2. MISTRAL 7B

Mistral 7B is a recent language model released by the Mistral AI team (Jiang et al., 2023) which has been shown to outperform Llama 2-13B on all benchmarks and Llama 1-34B on many benchmarks (Mistral AI team, 2023), and thus is considered the most powerful language models for its size to date. We experiment our scaled optimizers with this new language model on the GLUE (Wang et al., 2018) benchmark for natural language understanding problems. Table 2 shows the final fine-tuning results. Notably that our scaled optimizers outperform unscaled optimizers on all evaluation metrics, which demonstrates the effectiveness of gradient scaling for fine-tuning Mistral 7B model. See Appendix E.2 for training hyperparameter selection.

6.4. Diffusion Model Fine-Tuning

Diffusion models are now used for various image generation tasks and LoRA has been widely used for fine-tuning diffusion models. Here we start with the commonly used Stable Diffusion V1.5 model and show the effectiveness of applying our new preconditioner in LoRA fine-tuning for object generation. Then, we experiment with the Mix-of-Show model (Gu et al., 2023) which can generate high-quality face images. We observe that with gradient scaling, image generation becomes much more robust to learning rate changes, which is a reflection of the fact that our new preconditioner stabilizes the training process against learning rate variations. This has important practical benefits since learning rate choices can be crucial in image generation problems where small difference in learning rate can produce images of very different quality. This can be observed from Figures 1 and 3. Furthermore, it's widely observed that training loss is useless in monitoring image generation quality when training diffusion models. Thus a better optimizer which is

Method	GLUE									
	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	WNLI	Avg.
$\text{SGD}_{r=16}$	88.15	96.10	70.10	55.89	94.22	88.59	50.90	47.64	49.30	71.21
scaled GD (ours)$_{r=16}$	90.21	96.90	81.62	68.17	94.40	91.15	54.15	90.31	56.34	80.36
$\text{AdamW}_{r=16}$	89.86	96.79	88.48	71.05	94.42	91.24	90.61	90.42	81.69	88.28
scaled AdamW (ours)$_{r=16}$	**90.68**	**97.25**	**89.46**	**71.30**	**94.67**	**92.22**	**91.34**	**91.10**	**83.10**	**89.01**

Table 2. Scores for LoRA fine-tuning of 4-bit quantized Mistral 7B model on GLUE benchmark for Natural Language Understanding (NLU) challenges with different optimizers. Our scaled optimizers outperform unscaled optimizers on all evaluation metrics. See Section 6.3.2 for experimental details.

more robust to learning rate choices is very important. See Appendix F for experimental details for diffusion model fine-tuning tasks.

6.4.1. Object Generation

We build our object generation experiments on a popular stable diffusion fine-tuning repository (Ryu, 2023) with Stable Diffusion V1.5 as the base model. We follow the default settings there and tune both the U-Net and the text encoder where LoRA parameters are injected. For all experiments, we fix the U-Net fine-tuning learning rate as default value $1e-4$ which we find important for generating recognizable images. After fine-tuning on 6 images of a red vase titled "a photo of $\langle V_{\text{vase}}\rangle$", Figure 1 shows the generation results for prompt "a blue $\langle V_{\text{vase}}\rangle$". With large learning rate as $1e-2$ for text encoder fine-tuning, AdamW produces out-of-distribution results while our method produces satisfactory images. With default learning rate setting, AdamW still fails to capture the prompt information and generates only red vases. Instead, scaled AdamW with default learning rate is able to produce the desired blue vase. AdamW turns out to be able to generate the desired blue vase for learning rate value such as $1e-6$. See Appendix F.1 for other target object generation including chairs and dogs. Scaled AdamW improves AdamW for all experiments.

6.4.2. Face Generation

Face generation is a more challenging task compared to object generation and we thus switch to Mix-of-Show (Gu et al., 2023) variant of custom diffusion model which is originally designed for multi-concept LoRA and has been recognized to be able to generate high-quality face images. For better visualization of differences between different LoRA optimization methods, we turn off embedding fine-tuning and tune only text-encoders and U-Nets where LoRA factors are injected. We use 14 images of Potter provided in the original project repository where the character name is replaced with $\langle V_{\text{potter}}\rangle$ in captions of the training images. Figure 3 shows generation results for prompt "a pencil sketch of $\langle V_{\text{potter}}\rangle$" for various step sizes. Our method (scaled AdamW) generates visually better images more resemble a pencil sketch, which demonstrates its effectiveness in generating images of higher quality and also its robustness to learning rate changes. See Appendix F.2.1 for generation results for more prompts and also for the Hermoine character with different LoRA parameter fusion coefficients. See also Appendix F.2.2 for generation results including SGD and scaled GD methods for varying learning rates. Our observations are similar in all these generation results.

7. Convergence Theory

In this section, we further verify the superiority of scaled GD method applied to a reparameterized two-layer ReLU NN tuning problem, i.e., we show scaled GD method has convergence rate independent of data condition number of this specific problem and is thus advantageous compared to plain gradient descent. When scaled GD is applied to deep learning models, nonlinearities in such models typically render theoretical analysis intractable. To tackle this, we instead study an equivalent problem to the two-layer ReLU neural network. We first introduce the concept of hyperplane arrangement matrices. For a data matrix $X \in \mathbb{R}^{n\times d}$ and any arbitrary vector $u \in \mathbb{R}^d$, We consider the set of diagonal matrices

$$\mathcal{D} := \{\text{diag}(\mathbb{1}\{Xu \geq 0\})\},$$

which takes value 1 or 0 along the diagonals that indicate the set of possible arrangement activation patterns for the ReLU activation. Indeed, we can enumerate the set of sign patterns as $\mathcal{D} = \{D_i\}_{i=1}^P$ where P is bounded by

$$P \leq 2r\left(\frac{e(n-1)}{r}\right)^r$$

for $r = \text{rank}(X)$ (Pilanci & Ergen, 2020; Stanley et al., 2004). The two-layer ReLU model is equivalent to the problem below for squared loss through convexification under mild conditions[1] (Mishkin et al., 2022),

$$\min_{W_i} \frac{1}{2}\|\sum_{i=1}^P D_i X W_i - Y\|_F^2.$$

[1]The equivalence holds when the number of hidden neurons is greater than or equal to $2Pd$.

Figure 3. Generation results for prompt "a pencil sketch of $\langle V_{\text{potter}} \rangle$" by Mix-of-Show model with different optimizers and various learning rates. Our method (scaled AdamW) generates photos better capturing the prompt, i.e., a pencil sketch, and is more robust to learning rate choices. See Section 6.4.2 for experimental details.

Therefore, we base our analysis on fine-tuning the above model and show that the convergence rate of problem below with scaled GD method (Algorithm 2) has no dependence on condition number of the data matrix X. We focus on

$$\min_{A_i,B_i} \frac{1}{2}\|\sum_{i=1}^{P} D_i X(W_i + A_i B_i^T) - Y\|_F^2, \tag{3}$$

where $X \in \mathbb{R}^{n\times d}, A_i \in \mathbb{R}^{d\times r}, B_i \in \mathbb{R}^{c\times r}, Y \in \mathbb{R}^{n\times c}$. We consider the response model $Y = \sum_{i=1}^{P} D_i X(W_i + A_\star^i {B_\star^i}^T)$. Here, $X_\star^i = A_\star^i {B_\star^i}^T = U_\star^i \Sigma_\star^i {V_\star^i}^T$ are fixed and unknown matrices with $U_\star^i \Sigma_\star^i {V_\star^i}^T$ being the singular value decomposition of $X_\star^i$. Denote $F_\star^i = [A_\star^i, B_\star^i]^T$ and $F_t^i = [A_t^i, B_t^i]^T$ with A_t^i and B_t^i denote the value of (A_i, B_i) at t-th iteration. Let $\sigma_r(\cdot)$ denote the r-th largest singular value.

We first introduce the definition of Restricted Isometry Property (RIP) and illustrate the assumptions required for our theorem to hold,

Definition 7.1. (RIP (Recht et al., 2010)) The matrix $A \in \mathbb{R}^{n\times d}$ satisfies rank-r RIP with a constant $\delta_r \in [0, 1)$ if for all matrices $M \in \mathbb{R}^{d\times c}$ of rank at most r, the below holds,

$$(1 - \delta_r)\|M\|_F^2 \le \|AM\|_F^2 \le (1 + \delta_r)\|M\|_F^2.$$

Assumption 7.2. Suppose that $D_i X$ obeys the $2r$-RIP with a constant δ_{2r}^i for each i, and $\|X^T {D_i}^T D_j X\|_2 \le \min(\frac{\delta_{2r}^i \|X_\star^i\|_F}{P\|X_\star^j\|_F}, \frac{0.12}{7P(P+1)})$ for any $j \neq i$.

Note for matrix X with i.i.d Gaussian entries $\mathcal{N}(0, 1/d\|D_i\|_0)$, $D_i X$ satisfies RIP for a constant δ when $\|D_i\|_0$ is on the order of $r(d + c)/(d\delta^2)$. See (Recht et al., 2010) for other measurement ensembles satisfying the RIP property. Note also $\|X^T {D_i}^T D_j X\|_2 \le \|X^T X\|_2$ for all (i, j)'s. Thus bounding $\|X^T {D_i}^T D_j X\|_2$ amounts to bounding largest singular value of empirical covariance matrix. We consider a specific initialization strategy here which is an extension of spectral initialization for multiple terms as below,

Definition 7.3. (Extended Spectral Initialization) Let $A_0^i {B_0^i}^T$ be the best rank-r approximation of $(D_i X)^T (Y - \sum_{j=1}^{P} D_j X W_j)$ for each i.

Now, we are ready to state our main convergence result as follows,

Theorem 7.4. *Under Assumption 7.2 with $\delta_{2r}^i \le 0.01$ for each i. With extended spectral initialization described in Definition 7.3, $\|A_t^i {B_t^i}^T - X_\star^i\|_F \le 1.5 dist(F_t^i, F_\star^i)$. In addition, if the step size $0 < \eta \le 2/3$, then the $(t+1)$-th iteration F_{t+1}^i satisfies*

$$\max_i (dist(F_{t+1}^i, F_\star^i)) \le (1 - 0.5\eta) \max_i (dist(F_t^i, F_\star^i)).$$

Proof. See Appendix B. □

Our result mainly builds on results from (Tong et al., 2021b) and can be viewed as an extension of matrix sensing to ReLU neural networks.

8. Literature Review

Our work is closely related to low-rank matrix optimization and we briefly review some basic knowledge and related work in Section 8.1. Our work applies preconditioners for accelerating the LoRA fine-tuning process, which falls into preconditioning methods for PEFT, and related prior work there is discussed in Section 8.2 and Section 8.3.

8.1. Riemannian Optimization

Several recent studies have made theoretic contributions to the convergence rate of scaled GD method which employs the preconditioner (1). Specifically, in (Tong et al., 2021a;b),

the authors show local convergence of scaled GD method with better convergence rate independent of data condition number compared to plain gradient descent method for some classic low-rank matrix optimization problem including matrix sensing, robust PCA, etc. The authors of (Jia et al., 2023) show global convergence of scaled GD method with rate independent of data condition number for least squares matrix decomposition problem $\|AB^T - Y\|_F^2/2$. Different variants of scaled GD have been proposed and studied. In (Zhang et al., 2023c), the authors suggest to use $(A^TA + \lambda I)^{-1}$ and $(B^TB + \lambda I)^{-1}$ with some fixed $\lambda > 0$ in replace of (1) for tackling overparametrization and ill-conditioness in matrix sensing problems. In (Zhang et al., 2023b), the authors suggest using $(A^TA + \lambda_t I)^{-1}$ where $\lambda_{t+1} = \beta\lambda_t$ (similar change for B), i.e., using an exponentially decay regularization term. (Zhang et al., 2023c) proposes precGD method which sets $\lambda_t = \sqrt{f(A_tB_t^T)}$. (Tong et al., 2022; Ma et al., 2023) present extension of scaled GD method to tensor optimization problem. (Jia et al., 2023) analyzes AltScaledGD method which updates A and B alternatively and shows that such method has better convergence rate for larger step size.

8.2. Preconditioners in Deep Learning

Current deep learning training is dominated by gradient-based method which follows a descent direction to update parameters for decreasing objective value. For accelerating such training procedure, more advanced techniques such as Adagrad (Duchi et al., 2011) proposes to scale gradient based on their variance. Specifically, $G_t^{-1/2}$ is used as gradient preconditioner where G_t is accumulated outer product of historic subgradients. More practical optimizers such as Adam (Kingma & Ba, 2017) and AdamW (Loshchilov & Hutter, 2019) perform like a diagonal version of Adagrad and are the main training tools for most deep learning models in various fields. More recently, Shampoo (Gupta et al., 2018) has been proposed which uses a left preconditioner and a right preconditioner for a weight matrix. Shampoo is in spirit close to Adagrad but requires much less storage. In contrast with the preconditioners designed for accelerating optimization procedure for general deep learning models. We study a specific preconditioner designed for LoRA fine-tuning model which exploits its low rank matrix factorization property and borrows from Riemannian optimization knowledge.

8.3. PEFT Fine-Tuning Review

Current commonly-used deep learning models are growing larger and larger, making full fine-tuning for downstream tasks nearly impossible. A line of parameter-efficient fine-tuning methods emerges and has been used in various fields. These methods aim at achieving low fine-tuning loss with fewer trainable parameters. One popular PEFT method is LoRA (Hu et al., 2021), which proposes to add a low-rank adaptation to each existing weight matrix. By factorizing the update into two low-rank matrices, LoRA is able to achieve similar fine-tuning result as full fine-tuning with 10,000 times fewer parameters. LoRA has shown good performance in both language model fine-tuning and vision model fine-tuning. Variants of LoRA method involve DyLoRA (Valipour et al., 2023), IncreLoRA (Zhang et al., 2023a), and AdaLoRA (Zhang et al., 2023d), all focus on dynamically adjusting the rank hyperparameter. GLoRA (Chavan et al., 2023) generalizes LoRA by introducing a prompt module; Delta-LoRA (Zi et al., 2023) proposes to simultaneously update pretrained model weights by difference of LoRA weights. QLoRA (Dettmers et al., 2023) exploits quantized LoRA model which further reduces the model size. Besides such additive methods, there are also multiplicative PEFT methods such as the orthogonal fine-tuning method (OFT) (Qiu et al., 2023) and its variant BOFT (Liu et al., 2023).

Though LoRA has become very popular and different variants emerge, current LoRA training mainly exploits gradient based optimizers and we are unaware of any prior work studying acceleration of LoRA training given its special low-rank matrix factorization nature. Our work shows that by regrouping trainable parameters and applying an $r \times r$ preconditioner, the optimization procedure of LoRA can be significantly enhanced with negligible storage and runtime overhead.

9. Conclusion

In this work, we borrow tools from Riemannian optimization to enhance LoRA fine-tuning. Specifically, we study the application of a Riemannian gradient preconditioning method which introduces a new $r \times r$ preconditioner to LoRA fine-tuning procedure. Empirically, we observe that the gradient scaling boosts performance of both SGD and AdamW methods and theoretically we show that LoRA trained with preconditioned Adam method achieves stable feature learning under infinite-width NN setting while unpreconditioned training would require tuning learning rates for LoRA parameters separately. Prior to our work, theoretic convergence for the proposed gradient scaling scheme has only been established for classic low-rank matrix optimization problems and only with gradient descent method, we first time introduces it to deep learning regime considering the low-rank nature of LoRA model and reveals its superiority beyond SGD.

Acknowledgements

This work was supported in part by the National Science Foundation (NSF) under Grant DMS-2134248; in part by the NSF CAREER Award under Grant CCF-2236829; in

part by the U.S. Army Research Office Early Career Award under Grant W911NF-21-1-0242; and in part by the Office of Naval Research under Grant N00014-24-1-2164.

Impact Statement

This paper aims to advance the field of machine learning through innovative research. While our work holds significant potential for societal impact, we do not identify any specific consequences that needs to be highlighted here.

References

Candes, E. J., Li, X., Ma, Y., and Wright, J. Robust principal component analysis?, 2009.

Chavan, A., Liu, Z., Gupta, D., Xing, E., and Shen, Z. One-for-all: Generalized lora for parameter-efficient fine-tuning, 2023.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms, 2023.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61): 2121–2159, 2011.

Gal, R., Alaluf, Y., Atzmon, Y., Patashnik, O., Bermano, A. H., Chechik, G., and Cohen-Or, D. An image is worth one word: Personalizing text-to-image generation using textual inversion, 2022.

Gardent, C., Shimorina, A., Narayan, S., and Perez-Beltrachini, L. The WebNLG challenge: Generating text from RDF data. In Alonso, J. M., Bugarín, A., and Reiter, E. (eds.), *Proceedings of the 10th International Conference on Natural Language Generation*, pp. 124–133, Santiago de Compostela, Spain, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-3518. URL https://aclanthology.org/W17-3518.

Gu, Y., Wang, X., Wu, J. Z., Shi, Y., Chen, Y., Fan, Z., Xiao, W., Zhao, R., Chang, S., Wu, W., Ge, Y., Shan, Y., and Shou, M. Z. Mix-of-show: Decentralized low-rank adaptation for multi-concept customization of diffusion models, 2023.

Gupta, V., Koren, T., and Singer, Y. Shampoo: Preconditioned stochastic tensor optimization, 2018.

Hayou, S., Doucet, A., and Rousseau, J. On the impact of the activation function on deep neural networks training, 2019.

Hayou, S., Ghosh, N., and Yu, B. Lora+: Efficient low rank adaptation of large models, 2024.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models, 2021.

Jia, X., Wang, H., Peng, J., Feng, X., and Meng, D. Preconditioning matters: Fast global convergence of non-convex matrix factorization via scaled gradient descent. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b, 2023.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2017.

Labonne, M. Mistral fine-tuning example, 2024. URL https://towardsdatascience.com/fine-tune-a-mistral-7b-model\-with-direct-preference-optimization\-708042745aac. Accessed: 2024-01-25.

Liu, W., Qiu, Z., Feng, Y., Xiu, Y., Xue, Y., Yu, L., Feng, H., Liu, Z., Heo, J., Peng, S., Wen, Y., Black, M. J., Weller, A., and Schölkopf, B. Parameter-efficient orthogonal finetuning via butterfly factorization, 2023.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization, 2019.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps, 2022.

Ma, C., Xu, X., Tong, T., and Chi, Y. Provably accelerating ill-conditioned low-rank estimation via scaled gradient descent, even with overparameterization, 2023.

Mishkin, A., Sahiner, A., and Pilanci, M. Fast convex optimization for two-layer relu networks: Equivalent model classes and cone decompositions, 2022.

Mishra, B. and Sepulchre, R. Riemannian preconditioning. *SIAM Journal on Optimization*, 26(1):635–660, January 2016. ISSN 1095-7189. doi: 10.1137/140970860. URL http://dx.doi.org/10.1137/140970860.

Mishra, B., Apuroop, K. A., and Sepulchre, R. A riemannian geometry for low-rank matrix completion, 2012.

Mistral AI team. Mistral-7b announcement, 2023. URL https://mistral.ai/news/announcing-mistral-7b. Accessed: 2024-01-25.

Nan, L., Radev, D., Zhang, R., Rau, A., Sivaprasad, A., Hsieh, C., Tang, X., Vyas, A., Verma, N., Krishna, P., Liu, Y., Irwanto, N., Pan, J., Rahman, F., Zaidi, A., Mutuma, M., Tarabar, Y., Gupta, A., Yu, T., Tan, Y. C., Lin, X. V., Xiong, C., Socher, R., and Rajani, N. F. DART: Open-domain structured data record to text generation. In Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., and Zhou, Y. (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 432–447, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.37. URL https://aclanthology.org/2021.naacl-main.37.

Novikova, J., Dušek, O., and Rieser, V. The e2e dataset: New challenges for end-to-end generation, 2017.

Pilanci, M. and Ergen, T. Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks, 2020.

Qiu, Z., Liu, W., Feng, H., Xue, Y., Feng, Y., Liu, Z., Zhang, D., Weller, A., and Schölkopf, B. Controlling text-to-image diffusion by orthogonal finetuning, 2023.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019. URL https://api.semanticscholar.org/CorpusID:160025533.

Recht, B., Fazel, M., and Parrilo, P. A. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3): 471–501, January 2010. ISSN 1095-7200. doi: 10.1137/070697835. URL http://dx.doi.org/10.1137/070697835.

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

Ryu, S. Low-rank adaptation for fast text-to-image diffusion fine-tuning. https://github.com/cloneofsimo/lora/tree/master, 2023.

Schoenholz, S. S., Gilmer, J., Ganguli, S., and Sohl-Dickstein, J. Deep information propagation, 2017.

Stanley, R. P. et al. An introduction to hyperplane arrangements. *Geometric combinatorics*, 13(389-496):24, 2004.

Tong, T., Ma, C., and Chi, Y. Low-rank matrix recovery with scaled subgradient methods: Fast and robust convergence without the condition number. *IEEE Transactions on Signal Processing*, 69:2396–2409, 2021a. ISSN 1941-0476. doi: 10.1109/tsp.2021.3071560. URL http://dx.doi.org/10.1109/TSP.2021.3071560.

Tong, T., Ma, C., and Chi, Y. Accelerating ill-conditioned low-rank matrix estimation via scaled gradient descent, 2021b.

Tong, T., Ma, C., Prater-Bennette, A., Tripp, E., and Chi, Y. Scaling and scalability: Provable nonconvex low-rank tensor estimation from incomplete measurements, 2022.

Valipour, M., Rezagholizadeh, M., Kobyzev, I., and Ghodsi, A. Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation, 2023.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Linzen, T., Chrupała, G., and Alishahi, A. (eds.), *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://aclanthology.org/W18-5446.

Yang, G. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation, 2020.

Zhang, F. F., Li, L., Chen, J.-C., Jiang, Z., Wang, B., and Qian, Y. Increlora: Incremental parameter allocation method for parameter-efficient fine-tuning. *ArXiv*, abs/2308.12043, 2023a. URL https://api.semanticscholar.org/CorpusID:261076438.

Zhang, G., Chiu, H.-M., and Zhang, R. Y. Fast and minimax optimal estimation of low-rank matrices via non-convex gradient descent, 2023b.

Zhang, G., Fattahi, S., and Zhang, R. Y. Preconditioned gradient descent for overparameterized nonconvex burer–monteiro factorization with global optimality certification, 2023c.

Zhang, Q., Chen, M., Bukharin, A., He, P., Cheng, Y., Chen, W., and Zhao, T. Adaptive budget allocation for parameter-efficient fine-tuning, 2023d.

Zi, B., Qi, X., Wang, L., Wang, J., Wong, K.-F., and Zhang, L. Delta-lora: Fine-tuning high-rank parameters with the delta of low-rank matrices, 2023.

A. Details of Theorem 4.1

A.1. Assumptions and Technical Lemmas

Definition A.1. (Stable Feature Learning) Consider any general LoRA layer BAx with $B \in \mathbb{R}^{m\times r}$ and $A \in \mathbb{R}^{r\times n}$ being LoRA parameters. Denote $\Delta^t = B_tA_tx - B_{t-1}A_{t-1}x$ for fine-tuning step t. We say that LoRA model achieves Stable Feature Learning when $x, Ax, BAx \in \Theta(1)$ for all LoRA layers and $\Delta^t \in \Theta(1)$ for all fine-tuning step t.

Assumption A.2. We assume that the Adam gradient processing step satisfies $g_A^t x = \Theta(n)$ for all t where g_A^t is the normalized gradient of A in t-th iteration.

Explanation of Assumption A.2. We adopt the same assumption as Assumption 1 in (Hayou et al., 2024) where the authors provide a proof for Adam with no momentum, i.e., for SignSGD method. This assumption should hold for general Adam variants as long as the processed gradient preserves the sign(x) direction.

Lemma A.3. *For any matrix $A \in \mathbb{R}^{m\times n}$, where m being powers of n, such that A^TA is invertible and $\gamma[A_{ij}] = c$ for all (i, j), we have $\gamma[(A^TA)^{-1}] = -\gamma[\|a\|^2]$ with a being any column of A.*

Proof. First we note that $(A^TA)^{-1} = \text{adj}(A^TA)/\det(A^TA)$ and $\det(A^TA) = \Theta((mn^{2c})^n)$. Furthermore, by property of adjugate matrix,

$$\det(\text{adj}(A^TA)) = (\det(A^TA))^{n-1} = \Theta((mn^{2c})^{n(n-1)}),$$

from which we deduce

$$\text{adj}(A^TA) = \Theta((mn^{2c})^{n-1}).$$

Therefore,

$$(A^TA)^{-1} = \Theta((mn^{2c})^{-1}).$$

Note that $\|a\|^2 = \Theta(mn^{2c})$, we thus conclude $\gamma[(A^TA)^{-1}] = -\gamma[\|a\|^2]$, as desired. □

A.2. Statement and Proof of Theorem 4.1

Now, we state the formal version of our Theorem 4.1 below,

Theorem A.4. *(Stable Feature Learning (Formal)) Let g_A and g_B denote the processed gradient of A and B respectively. Consider LoRA parameters A and B trained with Adam scaled by preconditioner (1). Assume Assumption A.2 is satisfied with Adam gradient processing and $g_A, g_B \in \Theta(1)$ after the gradient processing. Further assume BAx has dimension of $\Theta(n)$. Then the LoRA model achieves stable feature learning with $\eta = \Theta(1)$. While for unscaled Adam, $\eta_A = \Theta(n^{-1})$ and $\eta_B = \Theta(1)$ are required for stable feature learning.*

Proof. We consider Gaussian initialization with $A_{ij} \sim \mathcal{N}(0, \sigma_a^2)$ and $B_{ij} \sim \mathcal{N}(0, \sigma_b^2)$. Conventionally, we want BA to be initialized as zero and Ax does not explode with NN width, thus we proceed with $\sigma_a^2 = \Theta(n^{-1})$ and $\sigma_b^2 = 0$. We first decompose the LoRA increment as

$$\begin{aligned}
\Delta^t &= B_tA_tx - B_{t-1}A_{t-1}x \\
&= (B_{t-1} - \eta g_B^{t-1}(A_{t-1}A_{t-1}^T)^{-1})(A_{t-1} - \eta(B_{t-1}^TB_{t-1})^{-1}g_A^{t-1})x - B_{t-1}A_{t-1}x \\
&= -\eta B_{t-1}(B_{t-1}^TB_{t-1})^{-1}g_A^{t-1}x - \eta g_B^{t-1}(A_{t-1}A_{t-1}^T)^{-1}A_{t-1}x + \eta^2 g_B^{t-1}(A_{t-1}A_{t-1}^T)^{-1}(B_{t-1}^TB_{t-1})^{-1}g_A^{t-1}x.
\end{aligned}$$

We write

$$\begin{cases}
\delta_t^1 = \eta B_{t-1}(B_{t-1}^TB_{t-1})^{-1}g_A^{t-1}x, \\
\delta_t^2 = \eta g_B^{t-1}(A_{t-1}A_{t-1}^T)^{-1}A_{t-1}x, \\
\delta_t^3 = \eta^2 g_B^{t-1}(A_{t-1}A_{t-1}^T)^{-1}(B_{t-1}^TB_{t-1})^{-1}g_A^{t-1}x.
\end{cases}$$

Following Assumption A.2, we know $g_A^{t-1}x \in \Theta(n)$, thus having $\delta_t^1, \delta_t^2, B_{t-1}A_{t-1}x \in \Theta(1)$ equates to

$$\begin{cases}
\gamma[\eta] + \gamma[B_{t-1}] + \gamma[(B_{t-1}^TB_{t-1})^{-1}] + 1 = 0, \\
\gamma[\eta] + \gamma[(A_{t-1}A_{t-1}^T)^{-1}] + \gamma[A_{t-1}x] = 0, \\
\gamma[B_{t-1}] + \gamma[A_{t-1}x] = 0.
\end{cases} \tag{4}$$

For gradient update, we have

$$B_t = B_{t-1} - \eta g_B^{t-1}(A_{t-1}A_{t-1}^T)^{-1}$$
$$A_t x = A_{t-1}x - \eta(B_{t-1}^T B_{t-1})^{-1} g_A^{t-1} x,$$

and thus

$$\gamma[B_t] = \max(\gamma[B_{t-1}], \gamma[\eta] + \gamma[(A_{t-1}A_{t-1}^T)^{-1}])$$
$$\gamma[A_t x] = \max(\gamma[A_{t-1}x], \gamma[\eta] + \gamma[(B_{t-1}^T B_{t-1})^{-1}] + 1).$$

Note $A_1 = A_0$ and thus $\gamma[A_1 x] = \gamma[A_0 x] = 0$. Furthermore, $\gamma[B_1] = \gamma[\eta] + \gamma[(A_0 A_0^T)^{-1}]$. Since $\gamma[\|a_0\|_2^2] = 0$ for any row a_0 of A_0, $\gamma[(A_0 A_0^T)^{-1}] = 0$ by Lemma A.3. Therefore $\gamma[B_1] = 0$. Since $\gamma(\|b_1\|_2^2) = 1$ for any column b_1 of B_1, $\gamma[(B_1^T B_1)^{-1}] = -1$ by Lemma A.3 and thus $\gamma[A_2 x] = 0$ by the above recursion. Since $A_2 = A_1 - \eta(B_1^T B_1)^{-1} g_A^1 = A_0 - \Theta(n^{-1}), \gamma[\|a_2\|^2] = 0$ for any row a_2 of A_2 and again by Lemma A.3 we know $\gamma[(A_2 A_2^T)^{-1}] = 0$. Therefore $\gamma[B_2] = 0$. The recursion persists and we know $\gamma[B_t] = \gamma[A_t x] = 0$ for all t. Since $\gamma[(B_{t-1}^T B_{t-1})^{-1}] = -1$ and $\gamma[(A_{t-1}A_{t-1}^T)^{-1}] = 0$, all equations in (4) are satisfied. One can check that $\delta_t^3 \in \Theta(1)$ and therefore stable feature learning is achieves with $\eta = \Theta(1)$. Theorem 1 in (Hayou et al., 2024) shows that $\eta_A = \Theta(n^{-1})$ and $\eta_B = \Theta(1)$ are required for unpreconditioned LoRA training to achieve stable feature learning.

A.3. Explanation of Different Learning Rates

Here we note that learning rate $\eta = \Theta(1)$ is required for Theorem 4.1 while learning rate $\eta = \Theta(n^{-1})$ is used for our toy example described in Section 4. This discrepancy arises from different settings being considered in these two regimes. Specifically, Theorem 4.1 deals with vector output, i.e., we assume BAx is of dimension $\Theta(n)$. This is core to our preconditioner since we then have $\gamma[(B_{t-1}^T B_{t-1})^{-1}] = -1$ from $\gamma[B_{t-1}] = 0$. For scalar output as considered in the toy example, when $\gamma[B_{t-1}] = 0$, we will have $\gamma[(B_{t-1}^T B_{t-1})^{-1}] = 0$ and thus fail to scale the statics correctly. Then one would wonder for scalar output, whether setting $\eta = \Theta(n^{-1})$ would be the correct choice as in the toy example. This is no longer true due to our Assumption A.2. In the toy example, we have $g_a^t = (f_t(x) - y)bx$ and ${g_a^t}^T x = \Theta(n)$ is not guaranteed since it also scales with $f_t(x)$ and b. Instead, Theorem 4.1 would hold for scalar output with $\eta_1 = \Theta(1)$ and $\eta_t = \Theta(n^{-1})$ for $t > 1$ which we do not include in the theorem statement for simplicity and one can deduce following our proof technique of Theorem 4.1. The takeaway is that for scalar output, our preconditioner can still achieve stable feature learning with same order of magnitude learning rate for both A and B though one may need to tune learning rates across iterations, which is the current convention of learning rate scheduling.

□

B. Proof of Theorem 7.4

Note problem (3) is equivalent to the problem below up to a change of labels,

$$\min_{A_i, B_i} \|\sum_{i=1}^{P} C_i A_i B_i^T - Y\|_F^2, \tag{5}$$

where $C_i \in \mathbb{R}^{n\times d}, A_i \in \mathbb{R}^{d\times r}, B_i \in \mathbb{R}^{c\times r}, Y \in \mathbb{R}^{n\times c}$. Consider $Y = \sum_{i=1}^P C_i A_\star^i {B_\star^i}^T$. Denote $X_\star^i = A_\star^i {B_\star^i}^T = U_\star^i \Sigma_\star^i {V_\star^i}^T$ where $U_\star^i \Sigma_\star^i {V_\star^i}^T$ is the singular value decomposition of $X_\star^i$. Denote $F_\star^i = [A_\star^i, B_\star^i]^T$. For any $F = [A, B]^T$, consider the following distance metric

$$\text{dist}^2(F, F_\star^i) = \inf_{Q\in GL(r)} \left\|(AQ - A_\star^i){\Sigma_\star^i}^{1/2}\right\|_F^2 + \left\|(BQ^{-T} - B_\star^i){\Sigma_\star^i}^{1/2}\right\|_F^2, \tag{6}$$

where $GL(r)$ denotes the set of invertible matrix in $\mathbb{R}^{r\times r}$. Let $\sigma_r(\cdot)$ denote the rth largest singular value and κ_i denote condition number of $X_\star^i$. Consider the following scaled GD step:

$$A_{t+1}^i = A_t^i - \eta C_i^T (\sum_j C_j A_t^j {B_t^j}^T - Y) B_t^i ({B_t^i}^T B_t^i)^{-1},$$

$$B_{t+1}^i = B_t^i - \eta (C_i^T (\sum_j C_j A_t^j {B_t^j}^T - Y))^T A_t^i ({A_t^i}^T A_t^i)^{-1}.$$

Before we begin the main proof, we need the following partial Frobenius norm which has been introduced in Section A.3 in (Tong et al., 2021b) with some important properties studied there.

Definition B.1. (Partial Frobenius norm) For any matrix X, its partial Frobenius norm of order r is given by l_2 norm of vectors composed by its top-r singular values,

$$\|X\|_{F,r} = \sqrt{\sum_{i=1}^{r} \sigma_i^2(X)}.$$

Now we start the proof by first proving some useful lemmas.

Lemma B.2. *Under Assumption 7.2, let $F_0^i = [A_0^i, B_0^i]^T$, then the extented spectral initialization in Definition 7.3 satisfies*

$$dist(F_0^i, F_\star^i) \leq 10\delta_{2r}^i \sqrt{r}\kappa_i \sigma_r(X_\star^i).$$

Proof. According to Lemma 11 in (Tong et al., 2021b), since $A_0^i {B_0^i}^T - X_\star^i$ has rank at most $2r$,

$$\begin{aligned}\text{dist}(F_0^i, F_\star^i) &\leq \sqrt{\sqrt{2}+1}\|A_0^i {B_0^i}^T - X_\star^i\|_F, \\ &\leq \sqrt{2(\sqrt{2}+1)}\|A_0^i {B_0^i}^T - X_\star^i\|_{F,r}.\end{aligned}$$

Since $A_0^i {B_0^i}^T$ is the best rank r approximation of $C_i^T Y = \sum_{j=1}^P C_i^T C_j X_\star^j$. Then

$$\begin{aligned}\|A_0^i {B_0^i}^T - X_\star^i\|_{F,r} &\leq \|\sum_{j=1}^P C_i^T C_j X_\star^j - A_0^i {B_0^i}^T\|_{F,r} + \|\sum_{j=1}^P C_i^T C_j X_\star^j - X_\star^i\|_{F,r}, \\ &\leq 2\|(C_i^T C_i - I)X_\star^i + \sum_{j\neq i} C_i^T C_j X_\star^j\|_{F,r}, \\ &\leq 2\delta_{2r}^i \|X_\star^i\|_F + 2\sum_{j\neq i}\|C_i^T C_j X_\star^j\|_F,\end{aligned}$$

where the last inequality follows Lemma 15 and inequality (50) in (Tong et al., 2021b). Therefore,

$$\begin{aligned}\text{dist}(F_0^i, F_\star^i) &\leq 5\delta_{2r}^i \|X_\star^i\|_F + 5\sum_{j\neq i}\|C_i^T C_j X_\star^j\|_F, \\ &\leq 10\delta_{2r}^i \|X_\star^i\|_F \leq 10\delta_{2r}^i \sqrt{r}\kappa_i \sigma_r(X_\star^i).\end{aligned}$$

□

Lemma B.3. *(Contraction) Under assumption 7.2 with $\delta_{2r}^i \leq 0.01$. If the t-th iterate satisfies dist$(F_t^i, F_\star^i) \leq 0.1\sigma_r(X_\star^i)$ where $F_t^i = [A_t^i, B_t^i]^T$, then $\|A_t^i {B_t^i}^T - X_\star^i\|_F \leq 1.5$dist$(F_t^i, F_\star^i)$. In addition, if the step size $0 < \eta \leq 2/3$, then the $(t+1)$-th iteration F_{t+1}^i satisfies*

$$\max(dist(F_{t+1}^i, F_\star^i)) \leq (1 - 0.5\eta)\max(dist(F_t^i, F_\star^i)).$$

Proof. We first show $\|A_t^i {B_t^i}^T - X_\star^i\|_F \leq 1.5\text{dist}(F_t^i, F_\star^i)$. According to Lemma 9 in (Tong et al., 2021b), we know Q_t^i, the optimal alignment matrix between F_t^i and $F_\star^i$, i.e., the optimal value of problem (5) with F replaced by F_t^i is attained at Q_t^i, exists, denote $A^i = A_t^i Q_t^i$, $B^i = B_t^i {Q_t^i}^{-T}$, $\triangle_A^i = A^i - A_\star^i$, $\triangle_B^i = B^i - B_\star^i$. By derivation (45) in (Tong et al., 2021b), we further know for $\epsilon = 0.1$,

$$\|\triangle_A^i {\Sigma_\star^i}^{-1/2}\|_2 \vee \|\triangle_B^i {\Sigma_\star^i}^{-1/2}\|_2 \leq \epsilon,$$

where $\vee$ denotes maximum. Note

$$\begin{aligned}
\|A_t^i B_t^{iT} - X_\star^i\|_F &= \|A^i B^{iT} - X_\star^i\|_F \\
&= \|\triangle_A^i B^{iT} + A_\star^i \triangle_B^{iT}\|_F \\
&= \|\triangle_A^i \triangle_B^{iT} + \triangle_A^i B_\star^{iT} + A_\star^i \triangle_B^{iT}\|_F \\
&\leq \|\triangle_A^i \triangle_B^{iT}\|_F + \|\triangle_A^i B_\star^{iT}\|_F + \|A_\star^i \triangle_B^{iT}\|_F \\
&= \|\triangle_A^i \Sigma_\star^{i\,1/2}\|_F + \|\triangle_B^i \Sigma_\star^{i\,1/2}\|_F + \|\triangle_A^i \triangle_B^{iT}\|_F \\
&\leq \|\triangle_A^i \Sigma_\star^{i\,1/2}\|_F + \|\triangle_B^i \Sigma_\star^{i\,1/2}\|_F + \\
&\quad \frac{1}{2}(\|\triangle_A^i \Sigma_\star^{i\,-1/2}\|_2 \vee \|\triangle_B^i \Sigma_\star^{i\,-1/2}\|_2)(\|\triangle_A^i \Sigma_\star^{i\,1/2}\|_F + \|\triangle_B^i \Sigma_\star^{i\,1/2}\|_F) \\
&\leq (1+\frac{\epsilon}{2})(\|\triangle_A^i \Sigma_\star^{i\,1/2}\|_F + \|\triangle_B^i \Sigma_\star^{i\,1/2}\|_F) \\
&\leq (1+\frac{\epsilon}{2})\sqrt{2}\text{dist}(F_t^i, F_\star^i) \leq 1.5\text{dist}(F_t^i, F_\star^i).
\end{aligned} \tag{7}$$

Note the second last inequality follows from $\text{dist}(F_t^i, F_\star^i) = \sqrt{\|\triangle_A^i \Sigma_\star^{i\,1/2}\|_F^2 + \|\triangle_B^i \Sigma_\star^{i\,1/2}\|_F^2}$. We then proceed to show the contraction of distance. By definition,

$$\text{dist}^2(F_{t+1}^i, F_\star^i) \leq \|(A_{t+1}^i Q_t^i - A_\star^i)\Sigma_\star^{i\,1/2}\|_F^2 + \|(B_{t+1}^i Q_t^{i-T} - B_\star^i)\Sigma_\star^{i\,1/2}\|_F^2.$$

Substitute the update rule for L_{t+1}^i we get

$$\begin{aligned}
(A_{t+1}^i Q_t^i - A_\star^i)\Sigma_\star^{i\,1/2} &= (A_t^i Q_t^i - \eta C_i^T(\sum_j C_j A_t^j B_t^{jT} - Y)B_t^i(B_t^{iT}B_t^i)^{-1}Q_t^i - A_\star^i)\Sigma_\star^{i\,1/2} \\
&= (\triangle_A^i - \eta C_i^T(\sum_j C_j A_t^j B_t^{jT} - Y)B_t^i(B_t^{iT}B_t^i)^{-1}Q_t^i)\Sigma_\star^{i\,1/2} \\
&= (\triangle_A^i - \eta C_i^T(\sum_j C_j A_t^j B_t^{jT} - Y)B^i(B^{iT}B^i)^{-1})\Sigma_\star^{i\,1/2} \\
&= (\triangle_A^i - \eta C_i^T C_i(A^i B^{iT} - X_\star^i)B^i(B^{iT}B^i)^{-1} - \eta C B^i(B^{iT}B^i)^{-1})\Sigma_\star^{i\,1/2} \\
&\text{where } C = \sum_{j\neq i} C_i^T C_j(A_t^j B_t^{jT} - X_\star^j), \\
&= (\triangle_A^i - \eta(A^i B^{iT} - X_\star^i)B^i(B^{iT}B^i)^{-1} - \eta(C_i^T C_i - I)(A^i B^{iT} - X_\star^i)B^i(B^{iT}B^i)^{-1} \\
&\quad - \eta C B^i(B^{iT}B^i)^{-1})\Sigma_\star^{i\,1/2} \\
&\text{since } A^i B^{iT} - X_\star^i = \triangle_A^i B^{iT} + A_\star^i \triangle_B^{iT}, \\
&= (\triangle_A^i - \eta\triangle_A^i - \eta A_\star^i \triangle_B^{iT} B^i(B^{iT}B^i)^{-1} - \eta(C_i^T C_i - I)(A^i B^{iT} - X_\star^i)B^i(B^{iT}B^i)^{-1} \\
&\quad - \eta C B^i(B^{iT}B^i)^{-1})\Sigma_\star^{i\,1/2} \\
&= (1-\eta)\triangle_A^i \Sigma_\star^{i\,1/2} - \eta A_\star^i \triangle_B^{iT} B^i(B^{iT}B^i)^{-1}\Sigma_\star^{i\,1/2} - \\
&\quad \eta(C_i^T C_i - I)(A^i B^{iT} - X_\star^i)B^i(B^{iT}B^i)^{-1}\Sigma_\star^{i\,1/2} - \eta C B^i(B^{iT}B^i)^{-1}\Sigma_\star^{i\,1/2}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\|(A_{t+1}^i Q_t^i - A_\star^i)\Sigma_\star^{i\,1/2}\|_F^2 &= \|(1-\eta)\triangle_A^i \Sigma_\star^{i\,1/2} - \eta A_\star^i \triangle_B^{iT} B^i(B^{iT}B^i)^{-1}\Sigma_\star^{i\,1/2}\|_F^2 \\
&\quad + \eta^2\|(C_i^T C_i - I)(A^i B^{iT} - X_\star^i)B^i(B^{iT}B^i)^{-1}\Sigma_\star^{i\,1/2} + C B^i(B^{iT}B^i)^{-1}\Sigma_\star^{i\,1/2}\|_F^2 \\
&\quad - 2\eta\text{tr}(((1-\eta)\Sigma_\star^{i\,1/2}\triangle_A^{iT} - \eta\Sigma_\star^{i\,1/2}(B^{iT}B^i)^{-1}B^{iT}\triangle_B^i A_\star^{iT}) \\
&\quad ((C_i^T C_i - I)(A^i B^{iT} - X_\star^i)B^i(B^{iT}B^i)^{-1}\Sigma_\star^{i\,1/2} + C B^i(B^{iT}B^i)^{-1}\Sigma_\star^{i\,1/2})).
\end{aligned} \tag{8}$$

Follow derivation (46) in (Tong et al., 2021b), we can bound

$$\begin{aligned}&\|(1-\eta)\triangle_A^i{\Sigma_\star^i}^{1/2}-\eta A_\star^i{\triangle_B^i}^T{B^i}^T{B^i}({B^i}^T{B^i})^{-1}{\Sigma_\star^i}^{1/2}\|_F^2\\&\leq\left((1-\eta)^2+\frac{2\epsilon\eta(1-\eta)}{1-\epsilon}\right)\|\triangle_A^i{\Sigma_\star^i}^{1/2}\|_F^2+\frac{\eta^2(2\epsilon+\epsilon^2)}{(1-\epsilon)^2}\|\triangle_B^i{\Sigma_\star^i}^{1/2}\|_F^2.\end{aligned}\tag{9}$$

Next, we want to bound

$$\begin{aligned}&\|(C_i^TC_i-I)(A^i{B^i}^T-X_\star^i)B^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2}+CB^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2}\|_F^2\\=&\|(C_i^TC_i-I)(A^i{B^i}^T-X_\star^i)B^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2}\|_F^2+\|CB^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2}\|_F^2\\&+2\text{tr}(((C_i^TC_i-I)(A^i{B^i}^T-X_\star^i)B^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2})^TCB^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2}).\end{aligned}$$

By the bound for $\mathfrak{S}_4$ in Lemma 1 in (Tong et al., 2021b), we can bound

$$\|(C_i^TC_i-I)(A^i{B^i}^T-X_\star^i)B^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2}\|_F^2\leq\frac{{\delta_{2r}^i}^2(2+\epsilon)^2}{2(1-\epsilon)^2}(\|\triangle_A^i{\Sigma_\star^i}^{1/2}\|_F^2+\|\triangle_B^i{\Sigma_\star^i}^{1/2}\|_F^2)$$

We now proceed to bound $\|CB^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2}\|_F^2$. According to Lemma 9 in (Tong et al., 2021b), we know Q_t^j exists, the optimal alignment matrix between F_t^j and $F_\star^j$ exists. Denote $A^j=A_t^jQ_t^j$, $B^j=B_t^j{Q_t^j}^{-T}$, $\triangle_A^j=A^j-A_\star^j$, $\triangle_B^j=B^j-B_\star^j$. Since

$$\begin{aligned}&\|C_i^TC_j(A_t^j{B_t^j}^T-X_\star^j)B^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2}\|_F\\&=\|C_i^TC_j(\triangle_A^j{\triangle_B^j}^T+\triangle_A^j{B_\star^j}^T+A_\star^j{\triangle_B^j}^T)B^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2}\|_F\\&\leq\|C_i^TC_j\|_2(\|\triangle_A^j{\triangle_B^j}^T\|_F+\|\triangle_A^j{B_\star^j}^T\|_F+\|A_\star^j{\triangle_B^j}^T\|_F)\|B^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2}\|_2\\&\text{by Lemma 12 in (Tong et al., 2021b) and (7),}\\&\leq\frac{(2+\epsilon)\|C_i^TC_j\|_2}{2(1-\epsilon)}(\|\triangle_A^j{\Sigma_\star^j}^{1/2}\|_F+\|\triangle_B^j{\Sigma_\star^j}^{1/2}\|_F).\end{aligned}\tag{10}$$

Thus

$$\begin{aligned}\|CB^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2}\|_F^2&\leq(P-1)\sum_{j\neq i}\|C_i^TC_j(A_t^j{B_t^j}^T-X_\star^j)B^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2}\|_F^2\\&\leq(P-1)\sum_{j\neq i}\frac{(2+\epsilon)^2\|C_i^TC_j\|_2^2}{2(1-\epsilon)^2}(\|\triangle_A^j{\Sigma_\star^j}^{1/2}\|_F^2+\|\triangle_B^j{\Sigma_\star^j}^{1/2}\|_F^2).\end{aligned}$$

Next we bound

$$\begin{aligned}&|\text{tr}(((C_i^TC_i-I)(A^i{B^i}^T-X_\star^i)B^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2})^TCB^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2})|\\&\leq\|B^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2}\|_2^2|\text{tr}(((C_i^TC_i-I)(A^i{B^i}^T-X_\star^i))^TC)|\\&\text{by Lemma 12 in (Tong et al., 2021b),}\\&\leq\frac{1}{(1-\epsilon)^2}|\text{tr}(((C_i^TC_i-I)(A^i{B^i}^T-X_\star^i))^TC)|\\&\text{by Lemma 17 in (Tong et al., 2021b), since } C_i \text{ is } 2r\text{-RIP,}\\&\leq\frac{\delta_{2r}^i}{(1-\epsilon)^2}\|C\|_F\|A^i{B^i}^T-X_\star^i\|_F\\&\text{by derivation in (10),}\\&\leq\sum_{j\neq i}\frac{(1+\frac{\epsilon}{2})^2\delta_{2r}^i}{(1-\epsilon)^2}\|C_i^TC_j\|_2(\|\triangle_A^j{\Sigma_\star^j}^{1/2}\|_F+\|\triangle_B^j{\Sigma_\star^j}^{1/2}\|_F)(\|\triangle_A^i{\Sigma_\star^i}^{1/2}\|_F+\|\triangle_B^i{\Sigma_\star^i}^{1/2}\|_F).\end{aligned}$$

To summarize,

$$\begin{aligned}
&\|(C_i^TC_i - I)(A^i{B^i}^T - X_\star^i)B^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2} + CB^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2}\|_F^2 \\
&\leq \frac{{\delta_{2r}^i}^2(2+\epsilon)^2}{2(1-\epsilon)^2}(\|\triangle_A^i{\Sigma_\star^i}^{1/2}\|_F^2 + \|\triangle_B^i{\Sigma_\star^i}^{1/2}\|_F^2) + (P-1)\sum_{j\neq i}\frac{\|C_i^TC_j\|_2^2(2+\epsilon)^2}{2(1-\epsilon)^2}(\|\triangle_A^j{\Sigma_\star^j}^{1/2}\|_F^2 + \|\triangle_B^j{\Sigma_\star^j}^{1/2}\|_F^2) \\
&\quad + \sum_{j\neq i}\frac{2(1+\frac{\epsilon}{2})^2\sigma_{2r}^i}{(1-\epsilon)^2}\|C_i^TC_j\|_2(\|\triangle_A^j{\Sigma_\star^j}^{1/2}\|_F + \|\triangle_B^j{\Sigma_\star^j}^{1/2}\|_F)(\|\triangle_A^i{\Sigma_\star^i}^{1/2}\|_F + \|\triangle_B^i{\Sigma_\star^i}^{1/2}\|_F).
\end{aligned} \tag{11}$$

Finally, we move on to bound

$$\begin{aligned}
&|\text{tr}(((1-\eta){\Sigma_\star^i}^{1/2}{\triangle_A^i}^T - \eta{\Sigma_\star^i}^{1/2}({B^i}^TB^i)^{-1}{B^i}^T\triangle_B^i{A_\star^i}^T)((C_i^TC_i - I)(A^i{B^i}^T - X_\star^i)B^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2} \\
&+ CB^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2}))| \\
&\leq |\text{tr}((1-\eta){\Sigma_\star^i}^{1/2}{\triangle_A^i}^T(C_i^TC_i - I)(A^i{B^i}^T - X_\star^i)B^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2})| \\
&\quad + |\text{tr}(\eta{\Sigma_\star^i}^{1/2}({B^i}^TB^i)^{-1}{B^i}^T\triangle_B^i{A_\star^i}^T(C_i^TC_i - I)(A^i{B^i}^T - X_\star^i)B^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2})| \\
&\quad + |\text{tr}((1-\eta){\Sigma_\star^i}^{1/2}{\triangle_A^i}^TCB^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2})| + |\text{tr}(\eta{\Sigma_\star^i}^{1/2}({B^i}^TB^i)^{-1}{B^i}^T\triangle_B^i{A_\star^i}^TCB^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2})|.
\end{aligned} \tag{12}$$

First notice by the bound for $|\mathfrak{S}_2|$ in Lemma 1 in (Tong et al., 2021b),

$$\begin{aligned}
&|\text{tr}((1-\eta){\Sigma_\star^i}^{1/2}{\triangle_A^i}^T(C_i^TC_i - I)(A^i{B^i}^T - X_\star^i)B^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2})| \\
&\leq \frac{(1-\eta)\delta_{2r}^i(2+\epsilon)}{2(1-\epsilon)}(\frac{3}{2}\|\triangle_A^i{\Sigma_\star^i}^{1/2}\|_F^2 + \frac{1}{2}\|\triangle_B^i{\Sigma_\star^i}^{1/2}\|_F^2).
\end{aligned} \tag{13}$$

Similarly, by the bound for $|\mathfrak{S}_3|$ in Lemma 1 in (Tong et al., 2021b),

$$\begin{aligned}
&|\text{tr}(\eta{\Sigma_\star^i}^{1/2}({B^i}^TB^i)^{-1}{B^i}^T\triangle_B^i{A_\star^i}^T(C_i^TC_i - I)(A^i{B^i}^T - X_\star^i)B^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2})| \\
&\leq \frac{\eta\delta_{2r}^i(2+\epsilon)}{2(1-\epsilon)^2}(\frac{3}{2}\|\triangle_B^i{\Sigma_\star^i}^{1/2}\|_F^2 + \frac{1}{2}\|\triangle_A^i{\Sigma_\star^i}^{1/2}\|_F^2).
\end{aligned} \tag{14}$$

Next, consider bounding

$$\begin{aligned}
&|\text{tr}((1-\eta){\Sigma_\star^i}^{1/2}{\triangle_A^i}^TCB^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2})| \\
&\leq (1-\eta)\sum_{j\neq i}|\text{tr}({\Sigma_\star^i}^{1/2}{\triangle_A^i}^TC_i^TC_j(A_t^j{B_t^j}^T - X_\star^j)B^i({B^i}^TB^i)^{-1}{\Sigma_\star^i}^{1/2})| \\
&\leq (1-\eta)\sum_{j\neq i}\|C_i^TC_j\|_2\|A_t^j{B_t^j}^T - X_\star^j\|_F\|B^i({B^i}^TB^i)^{-1}\Sigma_\star^i{\triangle_A^i}^T\|_F \\
&\text{by Lemma 12 in (Tong et al., 2021b),} \\
&\leq \frac{1-\eta}{1-\epsilon}\sum_{j\neq i}\|C_i^TC_j\|_2\|A_t^j{B_t^j}^T - X_\star^j\|_F\|\triangle_A^i{\Sigma_\star^i}^{1/2}\|_F \\
&\text{by derivation in (7),} \\
&\leq \frac{(2+\epsilon)(1-\eta)}{2(1-\epsilon)}\sum_{j\neq i}\|C_i^TC_j\|_2(\|\triangle_A^j{\Sigma_\star^j}^{1/2}\|_F + \|\triangle_B^j{\Sigma_\star^j}^{1/2}\|_F)\|\triangle_A^i{\Sigma_\star^i}^{1/2}\|_F,
\end{aligned} \tag{15}$$

and

$$
\begin{aligned}
&|\mathrm{tr}(\eta {\Sigma_\star^i}^{1/2}(B^{iT}B^i)^{-1}B^{iT}\triangle_B^i {A_\star^i}^T CB^i(B^{iT}B^i)^{-1}{\Sigma_\star^i}^{1/2})| \\
&= \eta|\mathrm{tr}(\sum_{j\neq i} C_i^T C_j(A_t^j {B_t^j}^T - X_\star^j)B^i(B^{iT}B^i)^{-1}\Sigma_\star^i(B^{iT}B^i)^{-1}B^{iT}\triangle_B^i {A_\star^i}^T)| \\
&\leq \eta\sum_{j\neq i}\|C_i^TC_j\|_2\|A_t^j{B_t^j}^T - X_\star^j\|_F\|B^i(B^{iT}B^i)^{-1}\Sigma_\star^i(B^{iT}B^i)^{-1}B^{iT}\triangle_B^i{A_\star^i}^T\|_F \\
&\leq \eta\sum_{j\neq i}\|C_i^TC_j\|_2\|A_t^j{B_t^j}^T - X_\star^j\|_F\|B^i(B^{iT}B^i)^{-1}{\Sigma_\star^i}^{1/2}\|_2^2\|\triangle_B^i{L_\star^i}^T\|_F \\
&\text{by Lemma 12 in (Tong et al., 2021b),} \\
&\leq \frac{\eta}{(1-\epsilon)^2}\sum_{j\neq i}\|C_i^TC_j\|_2\|A_t^j{B_t^j}^T - X_\star^j\|_F\|\triangle_B^i{A_\star^i}^T\|_F \\
&\text{by derivation in (7),} \\
&\leq \frac{\eta(2+\epsilon)}{2(1-\epsilon)^2}\sum_{j\neq i}\|C_i^TC_j\|_2(\|\triangle_A^j{\Sigma_\star^j}^{1/2}\|_F + \|\triangle_B^j{\Sigma_\star^j}^{1/2}\|_F)\|\triangle_B^i{A_\star^i}^T\|_F.
\end{aligned}
\tag{16}
$$

Combine (13), (14), (15), (16) to bound (12)

$$
\begin{aligned}
&|\mathrm{tr}(((1-\eta){\Sigma_\star^i}^{1/2}{\triangle_A^i}^T - \eta{\Sigma_\star^i}^{1/2}(B^{iT}B^i)^{-1}B^{iT}\triangle_B^i{A_\star^i}^T)((C_i^TC_i - I)(A^iB^{iT} - X_\star^i)B^i(B^{iT}B^i)^{-1}{\Sigma_\star^i}^{1/2} \\
&+ CB^i(B^{iT}B^i)^{-1}{\Sigma_\star^i}^{1/2}))| \\
&\leq \frac{(1-\eta)\delta_{2r}^i(2+\epsilon)}{2(1-\epsilon)}(\frac{3}{2}\|\triangle_A^i{\Sigma_\star^i}^{1/2}\|_F^2 + \frac{1}{2}\|\triangle_B^i{\Sigma_\star^i}^{1/2}\|_F^2) + \frac{\eta\delta_{2r}^i(2+\epsilon)}{2(1-\epsilon)^2}(\frac{3}{2}\|\triangle_B^i{\Sigma_\star^i}^{1/2}\|_F^2 + \frac{1}{2}\|\triangle_A^i{\Sigma_\star^i}^{1/2}\|_F^2) \\
&+ \frac{(2+\epsilon)(1-\eta)}{2(1-\epsilon)}\sum_{j\neq i}\|C_i^TC_j\|_2(\|\triangle_A^j{\Sigma_\star^j}^{1/2}\|_F + \|\triangle_B^j{\Sigma_\star^j}^{1/2}\|_F)\|\triangle_A^i{\Sigma_\star^i}^{1/2}\|_F \\
&+ \frac{\eta(2+\epsilon)}{2(1-\epsilon)^2}\sum_{j\neq i}\|C_i^TC_j\|_2(\|\triangle_A^j{\Sigma_\star^j}^{1/2}\|_F + \|\triangle_B^j{\Sigma_\star^j}^{1/2}\|_F)\|\triangle_B^i{L_\star^i}^T\|_F \\
&\leq \frac{(1-\eta)\delta_{2r}^i(2+\epsilon)}{2(1-\epsilon)}(\frac{3}{2}\|\triangle_A^i{\Sigma_\star^i}^{1/2}\|_F^2 + \frac{1}{2}\|\triangle_B^i{\Sigma_\star^i}^{1/2}\|_F^2) + \frac{\eta\delta_{2r}^i(2+\epsilon)}{2(1-\epsilon)^2}(\frac{3}{2}\|\triangle_B^i{\Sigma_\star^i}^{1/2}\|_F^2 + \frac{1}{2}\|\triangle_A^i{\Sigma_\star^i}^{1/2}\|_F^2) \\
&+ \frac{(2+\epsilon)(1-\eta)}{2(1-\epsilon)}\sum_{j\neq i}\|C_i^TC_j\|_2(\frac{1}{2}\|\triangle_A^j{\Sigma_\star^j}^{1/2}\|_F^2 + \frac{1}{2}\|\triangle_B^j{\Sigma_\star^j}^{1/2}\|_F^2 + \|\triangle_A^i{\Sigma_\star^i}^{1/2}\|_F^2) \\
&+ \frac{\eta(2+\epsilon)}{2(1-\epsilon)^2}\sum_{j\neq i}\|C_i^TC_j\|_2(\frac{1}{2}\|\triangle_A^j{\Sigma_\star^j}^{1/2}\|_F^2 + \frac{1}{2}\|\triangle_B^j{\Sigma_\star^j}^{1/2}\|_F^2 + \|\triangle_B^i{L_\star^i}^T\|_F^2).
\end{aligned}
\tag{17}
$$

Substituting bounds (9), (11), (17), we derive bound for (8) as

$$
\begin{aligned}
\|(A_{t+1}^iQ_t^i - A_\star^i){\Sigma_\star^i}^{1/2}\|_F^2 \leq\ & ((1-\eta)^2 + \frac{2\epsilon\eta(1-\eta)}{1-\epsilon})\|\triangle_A^i{\Sigma_\star^i}^{1/2}\|_F^2 + \frac{\eta^2(2\epsilon+\epsilon^2)}{(1-\epsilon)^2}\|\triangle_B^i{\Sigma_\star^i}^{1/2}\|_F^2 \\
&+ \eta^2(\frac{{\delta_{2r}^i}^2(2+\epsilon)^2}{2(1-\epsilon)^2}(\|\triangle_A^i{\Sigma_\star^i}^{1/2}\|_F^2 + \|\triangle_B^i{\Sigma_\star^i}^{1/2}\|_F^2) \\
&+ (P-1)\sum_{j\neq i}\frac{\|C_i^TC_j\|_2^2(2+\epsilon)^2}{2(1-\epsilon)^2}(\|\triangle_A^j{\Sigma_\star^j}^{1/2}\|_F^2 + \|\triangle_B^j{\Sigma_\star^j}^{1/2}\|_F^2) \\
&+ \sum_{j\neq i}\frac{2(1+\frac{\epsilon}{2})^2\delta_{2r}^i}{(1-\epsilon)^2}\|C_i^TC_j\|(\|\triangle_A^j{\Sigma_\star^j}^{1/2}\|_F + \|\triangle_B^j{\Sigma_\star^j}^{1/2}\|_F)(\|\triangle_A^i{\Sigma_\star^i}^{1/2}\|_F
\end{aligned}
$$

$$+ \|\triangle_B^i {\Sigma_\star^i}^{1/2}\|_F)) + 2\eta(\frac{(1-\eta)\delta_{2r}^i(2+\epsilon)}{2(1-\epsilon)}(\frac{3}{2}\|\triangle_A^i {\Sigma_\star^i}^{1/2}\|_F^2 + \frac{1}{2}\|\triangle_B^i {\Sigma_\star^i}^{1/2}\|_F^2)$$
$$+ \frac{\eta\delta_{2r}^i(2+\epsilon)}{2(1-\epsilon)^2}(\frac{3}{2}\|\triangle_B^i {\Sigma_\star^i}^{1/2}\|_F^2 + \frac{1}{2}\|\triangle_A^i {\Sigma_\star^i}^{1/2}\|_F^2)$$
$$+ \frac{(2+\epsilon)(1-\eta)}{2(1-\epsilon)}\sum_{j\neq i}\|C_i^T C_j\|_2(\frac{1}{2}\|\triangle_A^j {\Sigma_\star^j}^{1/2}\|_F^2 + \frac{1}{2}\|\triangle_B^j {\Sigma_\star^j}^{1/2}\|_F^2 + \|\triangle_A^i {\Sigma_\star^i}^{1/2}\|_F^2)$$
$$+ \frac{\eta(2+\epsilon)}{2(1-\epsilon)^2}\sum_{j\neq i}\|C_i^T C_j\|_2(\frac{1}{2}\|\triangle_A^j {\Sigma_\star^j}^{1/2}\|_F^2 + \frac{1}{2}\|\triangle_B^j {\Sigma_\star^j}^{1/2}\|_F^2 + \|\triangle_B^i {A_\star^i}^T\|_F^2)).$$

We similarly derive bound for $\|(B_{t+1}^i {Q_t^i}^{-T} - B_\star^i){\Sigma_\star^i}^{1/2}\|_F^2$. Since $\text{dist}^2(F_t^i, F_\star^i) = \text{tr}(\triangle_A^i \Sigma_\star^i {\triangle_A^i}^T) + \text{tr}(\triangle_B^i \Sigma_\star^i {\triangle_B^i}^T)$, we thus get

$$\|(A_{t+1}^i Q_t^i - A_\star^i){\Sigma_\star^i}^{1/2}\|_F^2 + \|(B_{t+1}^i {Q_t^i}^{-T} - B_\star^i){\Sigma_\star^i}^{1/2}\|_F^2$$
$$\leq ((1-\eta)^2 + \frac{2\epsilon\eta(1-\eta)}{1-\epsilon})(\|\triangle_A^i {\Sigma_\star^i}^{1/2}\|_F^2 + \|\triangle_B^i {\Sigma_\star^i}^{1/2}\|_F^2) + \frac{\eta^2(2\epsilon+\epsilon^2)}{(1-\epsilon)^2}(\|\triangle_B^i {\Sigma_\star^i}^{1/2}\|_F^2 + \|\triangle_A^i {\Sigma_\star^i}^{1/2}\|_F^2)$$
$$+ \eta^2(\frac{{\delta_{2r}^i}^2(2+\epsilon)^2}{2(1-\epsilon)^2}(2\|\triangle_A^i {\Sigma_\star^i}^{1/2}\|_F^2 + 2\|\triangle_B^i {\Sigma_\star^i}^{1/2}\|_F^2) + (P-1)\sum_{j\neq i}\frac{\|C_i^T C_j\|_2^2(2+\epsilon)^2}{2(1-\epsilon)^2}(2\|\triangle_A^j {\Sigma_\star^j}^{1/2}\|_F^2$$
$$+ 2\|\triangle_B^j {\Sigma_\star^j}^{1/2}\|_F^2) + \sum_{j\neq i}\frac{2(1+\frac{\epsilon}{2})^2\delta_{2r}^i}{(1-\epsilon)^2}\|C_i^T C_j\|(2\|\triangle_A^j {\Sigma_\star^j}^{1/2}\|_F + 2\|\triangle_B^j {\Sigma_\star^j}^{1/2}\|_F)(\|\triangle_A^i {\Sigma_\star^i}^{1/2}\|_F$$
$$+ \|\triangle_B^i {\Sigma_\star^i}^{1/2}\|_F)) + 2\eta(\frac{(1-\eta)\delta_{2r}^i(2+\epsilon)}{2(1-\epsilon)}(2\|\triangle_A^i {\Sigma_\star^i}^{1/2}\|_F^2 + 2\|\triangle_B^i {\Sigma_\star^i}^{1/2}\|_F^2)$$
$$+ \frac{\eta\delta_{2r}^i(2+\epsilon)}{2(1-\epsilon)^2}(2\|\triangle_B^i {\Sigma_\star^i}^{1/2}\|_F^2 + 2\|\triangle_A^i {\Sigma_\star^i}^{1/2}\|_F^2) + \frac{(2+\epsilon)(1-\eta)}{2(1-\epsilon)}\sum_{j\neq i}\|C_i^T C_j\|_2(\|\triangle_A^j {\Sigma_\star^j}^{1/2}\|_F^2$$
$$+ \|\triangle_B^j {\Sigma_\star^j}^{1/2}\|_F^2 + \|\triangle_A^i {\Sigma_\star^i}^{1/2}\|_F^2 + \|\triangle_B^i {\Sigma_\star^i}^{1/2}\|_F^2) + \frac{\eta(2+\epsilon)}{2(1-\epsilon)^2}\sum_{j\neq i}\|C_i^T C_j\|_2(\|\triangle_A^j {\Sigma_\star^j}^{1/2}\|_F^2 +$$
$$\|\triangle_B^j {\Sigma_\star^j}^{1/2}\|_F^2 + \|\triangle_B^i {A_\star^i}^T\|_F^2 + \|\triangle_A^i {A_\star^i}^T\|_F^2))$$
$$\leq ((1-\eta)^2 + \frac{2\epsilon\eta(1-\eta)}{1-\epsilon} + \frac{\eta^2(2\epsilon+\epsilon^2)}{(1-\epsilon)^2} + \frac{\eta^2{\delta_{2r}^i}^2(2+\epsilon)^2}{(1-\epsilon)^2})\text{dist}^2(F_t^i, F_\star^i)$$
$$+ \sum_{j\neq i}\frac{(P-1)\eta^2(2+\epsilon)^2\|C_i^T C_j\|_2^2}{(1-\epsilon)^2}\text{dist}^2(F_t^j, F_\star^j)$$
$$+ \sum_{j\neq i}\frac{8\eta^2(1+\frac{\epsilon}{2})^2\delta_{2r}^i}{(1-\epsilon)^2}\|C_i^T C_j\|\text{dist}(F_t^i, F_\star^i)\text{dist}(F_t^j, F_\star^j) + \frac{2\eta(1-\eta)\delta_{2r}^i(2+\epsilon)}{(1-\epsilon)}\text{dist}^2(F_t^i, F_\star^i)$$
$$+ \frac{2\eta^2\delta_{2r}^i(2+\epsilon)}{(1-\epsilon)^2}\text{dist}^2(F_t^i, F_\star^i) + \sum_{j\neq i}\frac{\eta(2+\epsilon)(1-\eta)}{1-\epsilon}\|C_i^T C_j\|_2\text{dist}^2(F_t^j, F_\star^j)$$
$$+ \frac{\eta(2+\epsilon)(1-\eta)\sum_{j\neq i}\|C_i^T C_j\|_2}{(1-\epsilon)}\text{dist}^2(F_t^i, F_\star^i) + \sum_{j\neq i}\frac{\eta^2(2+\epsilon)}{(1-\epsilon)^2}\|C_i^T C_j\|_2\text{dist}^2(F_t^j, F_\star^j)$$
$$+ \sum_{j\neq i}\frac{\eta^2(2+\epsilon)\|C_i^T C_j\|_2}{(1-\epsilon)^2}\text{dist}^2(F_t^i, F_\star^i)$$
$$= ((1-\eta)^2 + \frac{2\epsilon\eta(1-\eta)}{1-\epsilon} + \frac{\eta^2(2\epsilon+\epsilon^2)}{(1-\epsilon)^2} + \frac{\eta^2{\delta_{2r}^i}^2(2+\epsilon)^2}{(1-\epsilon)^2} + \frac{2\eta(1-\eta)\delta_{2r}^i(2+\epsilon)}{(1-\epsilon)} + \frac{2\eta^2\delta_{2r}^i(2+\epsilon)}{(1-\epsilon)^2}$$
$$+ \frac{\eta(2+\epsilon)(1-\eta)\sum_{j\neq i}\|C_i^T C_j\|_2}{(1-\epsilon)} + \frac{\eta^2(2+\epsilon)\sum_{j\neq i}\|C_i^T C_j\|_2}{(1-\epsilon)^2})\text{dist}^2(F_t^i, F_\star^i)$$

$$+\sum_{j\neq i}\Big(\frac{(P-1)\eta^2\|C_i^TC_j\|^2(2+\epsilon)^2}{(1-\epsilon)^2}+\frac{\eta(2+\epsilon)(1-\eta)\|C_i^TC_j\|_2}{(1-\epsilon)}+\frac{\eta^2(2+\epsilon)\|C_i^TC_j\|_2}{(1-\epsilon)^2}\Big)\text{dist}^2(F_t^j,F_\star^j)$$

$$+\sum_{j\neq i}\frac{8\eta^2(1+\frac{\epsilon}{2})^2\delta_{2r}^i}{(1-\epsilon)^2}\|C_i^TC_j\|_2\text{dist}(F_t^i,F_\star^i)\text{dist}(F_t^j,F_\star^j)$$

Let σ_{ij} denote $\|C_i^TC_j\|_2$,

$$\leq\Big((1-0.6\eta)^2+\frac{7}{3}\sum_{j\neq i}\sigma_{ij}\eta+\frac{7}{27}\sum_{j\neq i}\sigma_{ij}\eta^2\Big)\text{dist}^2(F_t^i,F_\star^i)+\sum_{j\neq i}\Big(\frac{7}{3}\sigma_{ij}\eta+\Big(\frac{7}{27}\sigma_{ij}+\frac{49}{9}(P-1)\sigma_{ij}^2\Big)\eta^2\Big)\text{dist}^2(F_t^j,F_\star^j)$$

$$+\sum_{j\neq i}0.3\sigma_{ij}\eta^2\text{dist}(F_t^i,F_\star^i)\text{dist}(F_t^j,F_\star^j)$$

Let $\sigma=\max(\sigma_{ij})$ over all j, when $\sigma,\eta\leq 1$,

$$\leq\big((1-0.6\eta)^2+3(P-1)\sigma\eta\big)\text{dist}^2(F_t^i,F_\star^i)+\sum_{j\neq i}\big(3\sigma+6(P-1)\sigma^2\big)\eta\text{dist}^2(F_t^j,F_\star^j)$$

$$+\sum_{j\neq i}0.3\sigma\eta\text{dist}(F_t^i,F_\star^i)\text{dist}(F_t^j,F_\star^j)$$

WLOG assume $\text{dist}^2(F_t^k,F_\star^k)\geq\text{dist}^2(F_t^j,F_\star^j)$ for any $j\neq k$. Then when $\sigma\leq\min(1,\frac{0.12}{7P(P+1)})$,

$$\big((1-0.6\eta)^2+3(P-1)\sigma\eta\big)\text{dist}^2(F_t^k,F_\star^k)+\sum_{j\neq i}\big(3\sigma+6(P-1)\sigma^2\big)\eta\text{dist}^2(F_t^k,F_\star^k)+\sum_{j\neq i}0.3\sigma\eta\text{dist}^2(F_t^k,F_\star^k)$$

$$\leq(1-0.5\eta)^2\text{dist}^2(F_t^k,F_\star^k).$$

Therefore,

$$\max_i(\text{dist}^2(F_{t+1}^i,F_\star^i))\leq(1-0.5\eta)^2\max_i(\text{dist}^2(F_t^i,F_\star^i)).$$

□

The proof of Theorem 7.4 is a simple combination of Lemma B.2 and Lemma B.3. We build on (Tong et al., 2021b) for our proof. Note in (Jia et al., 2023), the authors provide a proof for global convergence for scaled GD method for least squares matrix decomposition problem. Since the proof detail there is closely tailed to the objective, we stick to the local convergence proof in (Tong et al., 2021b) which more resembles the problem we are interested in.

C. SGD with Gradient Scaling

Algorithm 2 Pseudocode of scaled GD in PyTorch.

```
# group trainable parameters into LoRA pairs in train.py
for LoRA_A, LoRA_B in pairwise(trainable_parameter):
    param_groups.append({"params":  [LoRA_A,LoRA_B], "lr":  learning_rate})

# apply preconditioner in optimizer.py
for group in param_groups:
     A, B = group["params"]
     dA, dB = group["params"].grad
     # precondition gradients
     dA_scaled =inverse(B.T@B+delta*torch.eye(r)).mm(dA)
     dB_scaled =dB.mm(inverse(A@A.T+delta*torch.eye(r)))
     # update parameters
     A.add_(dA_scaled, -group['lr'])
     B.add_(dB_scaled, -group['lr'])
```

`pairwise`: read every two elements in a list

D. More on Runtime Comparison

Figure 4 shows runtime comparison for fine-tuning GPT-2 model with LoRA trained with differernt optimizers, $r=256$ is adopted. Note though the runtime gap between scaled optimizers and unscaled ones increases compared to $r=4$ shown in Section 6.2, the increment is still marginal.

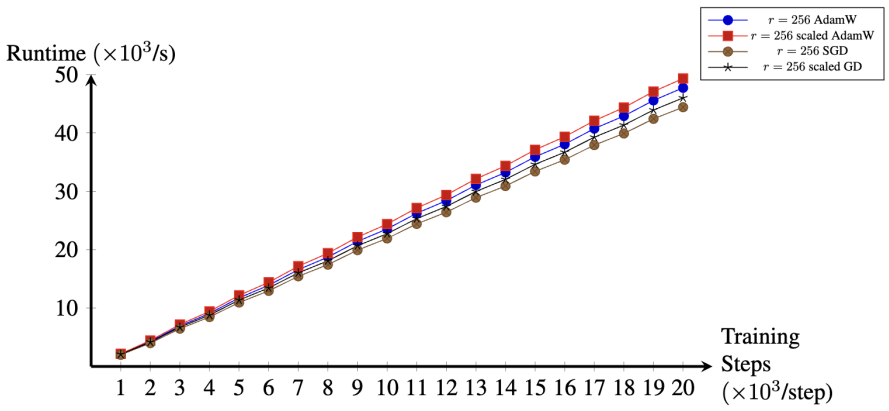

Figure 4. Runtime for LoRA fine-tuning GPT-2 medium model with rank $r = 256$ with different optimizers. Our scaled methods introduce marginal runtime overhead and train as fast as unscaled methods. See Section 6.3.1 for experimental details.

E. Supplementary Experiments for Language Models

E.1. GPT-2

E.1.1. Experimental Results for Different LoRA Ranks

For experiments for varying LoRA ranks, we follow exact the same setting as in original LoRA project (Hu et al., 2021). We experiment with medium-size GPT-2 (Radford et al., 2019) model with hyperparameters listed in Table 3, where the learning rates for different methods are individually tuned by grid search except for AdamW, which we follow default setting in LoRA report (Hu et al., 2021). We train with a linear learning rate schedule for 5 epochs. We note that we also tune hyperparameters β_1, β_2 for AdamW-type methods and find that lower β_1, β_2 values are beneficial to scaled AdamW method. See Table 4 for experimental results. We test with LoRA rank $r = 1, 4, 8$ and method with scaled gradient always performs better than its non-scaled gradient counterpart for all ranks and on most evaluation metrics. We set regularization factor to be $\sigma = 1e - 6$.

Method	SGD			scaled GD			AdamW			scaled AdamW		
Rank	1	4	8	1	4	8	1	4	8	1	4	8
	Training											
Weight decay	0.01											
Dropout Prob	0.1											
Batch Size	8											
# Epoch	5											
Warmup Steps	500											
LR Scheduler	Linear											
Label Smooth	0.1											
LR (tuned, $\times 10^{-3}$)	60	90	/	20	30	/	0.2			0.5	0.8	2
AdamW β_1	0.9									0.7		
AdamW β_2	0.999									0.8		
LoRA α	32											
	Inference											
Beam Size	10											
Length Penalty	0.8											
No Repeat Ngram Size	4											

Table 3. Hyperparameters for GPT-2 model fine-tuning

Method	Rank	E2E				
		BLEU	NIST	MET	ROUGE-L	CIDEr
SGD	1	35.9	5.09	25.3	46.2	0.48
scaled GD (ours)	1	68.2	8.65	45.7	69.6	2.44
AdamW	1	69.9	8.80	**46.5**	71.4	2.48
scaled AdamW (ours)	1	**70.1**	**8.82**	**46.5**	**71.7**	**2.51**
SGD	4	66.6	8.54	44.2	68.2	2.32
scaled GD (ours)	4	69.2	8.71	46.3	70.9	2.48
AdamW	4	68.9	8.69	46.5	71.3	2.51
scaled AdamW (ours)	4	**69.6**	**8.77**	**46.6**	**71.8**	**2.52**
SGD	8	65.8	8.46	43.5	68.7	2.33
scaled GD (ours)	8	69.6	8.78	46.4	70.8	2.48
AdamW	8	69.6	8.74	**46.7**	**71.8**	**2.53**
scaled AdamW (ours)	8	**70.1**	**8.82**	46.6	**71.8**	**2.53**

Table 4. Experiments for GPT-2 medium model on E2E NLG challenge with different LoRA ranks. Our scaled optimizers outperform unscaled optimizers for all LoRA ranks being tested and on most evaluation metrics. Moreover, scaled GD method behaves close to AdamW method. See Appendix E.1.1 for experimental details.

E.1.2. Experimental Results for Different Model Sizes

For GPT-2 model, we also experiment with different model sizes. LoRA rank r is fixed to be 4. We use the same training hyperparameters as listed in Table 3. See Table 5 for final scores. Note our scaled gradient methods always outperform their unscaled gradient counterparts for different model sizes and on most evaluation metrics, which shows the superiority of the introduced preconditioner. Furthermore, there is usually significant performance gap between SGD method and AdamW method while our scaled GD method is able to obtain scores comparable to AdamW without requiring momentum terms. Our scaled GD method indeed closes the gap between SGD and AdamW.

Method	Model	# Trainable Parameters	E2E				
			BLEU	NIST	MET	ROUGE-L	CIDEr
SGD	GPT-2 S	0.15M	54.8	4.56	34.0	63.3	1.29
scaled GD (ours)	GPT-2 S	0.15M	68.5	8.72	45.5	69.4	2.40
AdamW	GPT-2 S	0.15M	69.1	8.75	46.0	70.5	2.47
scaled AdamW (ours)	GPT-2 S	0.15M	**69.5**	**8.80**	**46.2**	**70.9**	**2.48**
SGD	GPT-2 M	0.39M	66.6	8.54	44.2	68.2	2.32
scaled GD (ours)	GPT-2 M	0.39M	69.2	8.71	46.3	70.9	2.48
AdamW	GPT-2 M	0.39M	68.9	8.69	46.5	71.3	2.51
scaled AdamW (ours)	GPT-2 M	0.39M	**69.6**	**8.77**	**46.6**	**71.8**	**2.52**

Table 5. Experiments for GPT-2 models with different sizes with LoRA rank $r = 4$ on E2E NLG challenge. Our scaled optimizers behave better than unscaled ones and scaled GD obtains scores comparable to AdamW. See Appendix E.1.2 for experimental details.

E.1.3. Experimental Results for Different Datasets

We experiment with also WebNLG (Gardent et al., 2017) and DART (Nan et al., 2021) datasets with GPT-2 medium-size model with LoRA rank $r = 4$. We use exactly the same training hyperparameters as listed in Table 3. WebNLG is a popular dataset for data-to-text evaluation introduced by (Gardent et al., 2017) which includes 22K examples from 14 distinct categories. Among these categories, five categories are presented only at test time and thus the evaluation is divided into "seen" (S), "unseen" (U), "all" (A) three types depending on whether the five categories are included in test time or not. DART is another data-to-text dataset introduced by (Nan et al., 2021) which involves 82K examples. The evaluation metrics being used are BLEU, MET, and TER with higher scores being better for first two metrics and the lower the better for TER. The experimental result is presented in Table 6 and we observe that scaled GD significantly improves SGD performance for both datasets on all evaluation metrics, and so does scaled AdamW for AdamW.

Method	DART			WebNLG								
	BLEU↑	MET↑	TER↓	BLEU↑			MET↑			TER↓		
				U	S	A	U	S	A	U	S	A
SGD	43.2	.36	.50	45.5	58.1	52.4	.36	.42	.39	**.45**	.36	.40
scaled GD (ours)	46.1	.38	.48	46.3	61.7	54.8	.37	.44	.41	**.45**	.34	.39
AdamW	47.1	.38	**.47**	45.0	64.1	55.5	**.38**	**.45**	**.42**	.47	**.32**	.39
scaled AdamW (ours)	**47.9**	**.39**	**.47**	**46.8**	**64.2**	**56.3**	**.38**	**.45**	**.42**	.46	**.32**	**.38**

Table 6. Experiments for GPT-2 medium model on NLG challenge with DART dataset and WebNLG dataset. Our scaled optimizers improve unscaled optimizers uniformly on all evaluation metrics for both datasets. See Appendix E.1.3 for experiment, datasets, and metric details.

E.2. Mistral 7B

Mistral 7B is a pretty new model up-to-date and there is no well-established code base we can follow. For quicker training, we use 4-bit quantized version of Mistral 7B V0.1 as our base model and LoRA factors are injected to each linear layer with rank $r = 16$. We train for 5 total epochs with batch size 8. For fine-tuning 4-bit quantized Mistral 7B V0.1 model for GLUE benchmark, we exploit HuggingFace transformers trainer class. All training arguments are default there except for the batch size, training epoch, and optimizer-related settings which are customized for our experiments. See Table 7 for training hyperparameter choices. The β's and ϵ for AdamW-type methods are defaultly used in original LoRA project (Hu et al., 2021). For learning rate choices, we view that values ranging from $2e-5$ to $2e-4$ have been empirically used for fine-tuning Mistral 7B for other tasks. We follow (Labonne, 2024) and set $lr = 5e-5$ for AdamW-type methods. For larger datasets including mnli, qnli, and qqp, $lr = 5e-5$ results in NaN loss thus we tune learning rate individually with grid search for each method. Since SGD has never been used for training Mistral 7B, we empirically find $lr = 5e-3$ to be a reasonable learning rate. For larger datasets including mnli, qnli, and qqp, we still tune the learning rate with grid search. Learning rate scheduler, warmup steps, warmup ratios, and max grad norm are all default in HuggingFace trainer class. Weight decay value 0.01 is what has been used in original LoRA project for GPT-2 fine-tuning. We use the same LoRA-related parameters and model quantization configuration as in (Labonne, 2024).

Method	SGD	scaled GD	AdamW	scaled AdamW
train batch size	8			
seed (default)	42			
AdamW (β_1, β_2)	/		$(0.9, 0.999)$	
AdamW ϵ	/		$1e-6$	
lr	$5e-3$		$5e-5$	
lr (mnli)	$5e-3$	$1e-3$	$5e-6$	$3e-5$
lr (qqp)	$5e-3$	$4e-3$	$5e-6$	$3e-5$
lr (qnli)	$5e-3$	$9e-4$	$1e-5$	
lr scheduler	linear			
num epoch	5			
warmup steps & warmup ratios	0			
weight decay	0.01			
max grad norm	1			
LoRA rank	16			
LoRA α	16			
LoRA dropout	0.05			
Load in 4-bit	True			
4bit quantization type	nf4			
4bit dtype	bfloat16			

Table 7. Hyperparameters for Mistral 7B model fine-tuning

F. Supplementary Experiments for Diffusion Models

F.1. Stable Diffusion

For our object generation experiment, we follow the popular custom diffusion repository (Ryu, 2023). We follow all default settings for both training and inference steps except for the optimizer component. We use "a photo of $\langle V_{\text{object}} \rangle$" as all object image captions. In the original repository, AdamW is used as default optimizer with learning rate $5e-5$ for text-encoder tuning and $1e-4$ for U-Net tuning. The pretrained model being used is Stable Diffusion V1.5 (Rombach et al., 2022). For training procedure, we use constant learning rate scheduler with zero learning rate warmup steps, which is the default setting. Max training step is set to 4000. For sampling procedure, we set number of inference steps to be 50 and guidance scale to be 7, which is used as default in the experiment notebook provided. Figure 5 shows generation results for a yellow chair with training images containing the target chair in color blue. AdamW is able to generate the target yellow chair only for learning rate $1e-6$ while our method generates desired images for all learning rates being tested. Figure 6 shows generation results for dog object. Our method generates images better capturing the prompt, i.e, a dog wearing a hat. For large learning rate $1e-2$, AdamW only generates black images while no black images are observed for scaled AdamW generation, which again verifies the robustness of our scaled optimizers.

Figure 5. Generation results for prompt "a yellow $\langle V_{\text{chair}} \rangle$" after fine-tuning on 5 blue chair images of the Stable Diffusion V1.5 model. We vary text-encoder learning rates with U-Net learning rate fixed to default value $1e-4$. No black images are observed for our method's generation and AdamW generates only black images for large learning rates. Our method (scaled AdamW) generates photos better capturing the prompt and is more robust to learning rate changes. See Appendix F.1 for experimental details.

Figure 6. Generation results for prompt "$\langle V_{\text{dog}} \rangle$ wearing a hat" after fine-tuning on 5 dog images of the Stable Diffusion V1.5 model. See Appendix F.1 for experimental details. Our method (scaled AdamW) generates images better capturing the prompt, i.e., a dog wearing a hat, and is more robust to learning rate changes.

F.2. Mix-of-Show

For our experiment for face generation tasks, we base our experiments on Mix-of-Show (Gu et al., 2023) repository which we find able to generate high-quality face images and is better for visualization comparison between different optimization

methods we consider. We follow default settings for training and inference in (Gu et al., 2023) with the exception that we turn off embedding tuning and only tune the text encoder and U-Net fraction where LoRA parameters are injected. The reason is that we find with embedding tuning, the effect of LoRA parameter is restricted and thus does no good to our comparison. In Mix-of-Show, Chilloutmix[2] is used as pretrained model. LoRA rank is set to 4. DMP-Solver (Lu et al., 2022) is employed for sampling. See (Gu et al., 2023) for more discussion on experimental details. Here we first fix step size $5e-4$ for both text encoder tuning and U-Net tuning and compare AdamW versus scaled AdamW with different LoRA parameter fusion coefficients. The default step size value used for Mix-of-Show is $1e-5$ and $1e-4$ for text encoder tuning and U-Net tuning respectively and default optimizer is AdamW. See Section F.2.1 for experimental results for different LoRA parameter fusion coefficients. Our scaled AdamW optimizer generates visually better images compared to AdamW optimizer. We then test with varying step size settings and demonstrate that our scaled gradient method is more robust to step size changes in Section F.2.2.

F.2.1. AdamW vs. scaled AdamW with Varying LoRA Parameter Fusion Coefficients

We experiment with Potter character and Hermione character. For Potter character, we use 14 Potter images for training LoRA parameters with the character name replace by special token $\langle V_{\text{potter}} \rangle$ as what has been introduced in textual inversion (Gal et al., 2022), then in the sampling procedure, we use prompt with special character $\langle V_{\text{potter}} \rangle$ for generating images involving Potter character. Figure 7, 8 and 9 show the generation results for three different prompts. The above two rows are for AdamW optimizer with the first row having LoRA parameter fusion coefficient 0.7 and second row 1 and the third and fourth rows correspond to scaled AdamW generation. LoRA parameter fusion coefficient represents α in $W = W_0 + \alpha AB^T$ when merging LoRA weights. We observe that our scaled AdamW method is able to generate higher quality images compared to unscaled version for both $\alpha = 0.7$ and $\alpha = 1$. For Hermione character, we train with 15 photos of Hermione following procedure described before. Figure 10 and 11 show the generation results. Still, our scaled AdamW optimizer produces higher quality images compared to AdamW optimizer.

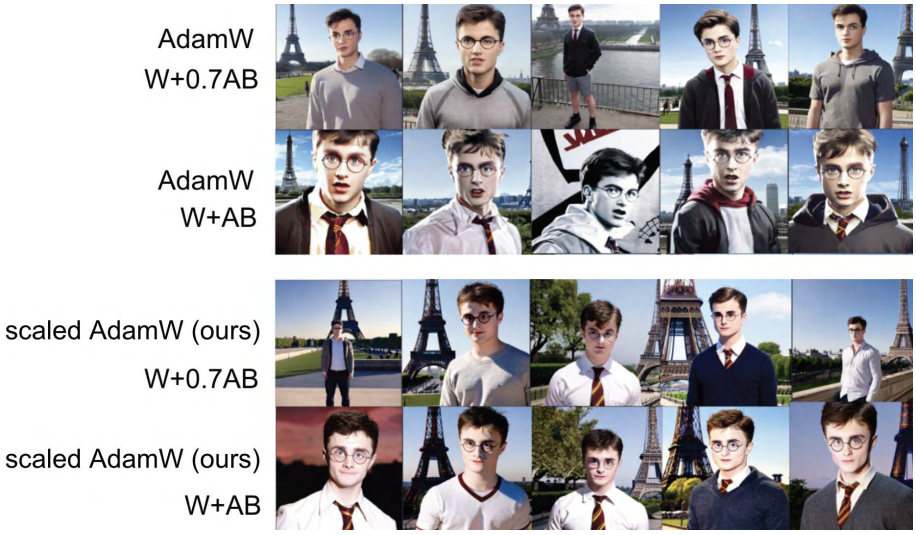

Figure 7. Generation results for prompt "a $\langle V_{\text{potter}} \rangle$ in front of eiffel tower" after fine-tuning on 14 Potter images. The above two rows are from AdamW optimizer and the bottom two rows are from our scaled AdamW optimizer. The first and third rows correspond to LoRA parameter fusion coefficient 0.7 and the second and fourth rows correspond to LoRA parameter fusion coefficient 1.0. Our scaled AdamW method generates images of higher quality compared to AdamW. See Appendix F.2.1 for experimental details.

F.2.2. Varying Step Sizes

Here we test with varying step sizes. Note the Mix-of-Show repository uses AdamW as default optimizer with learning rate $1e-5$ for text-encoder tuning and $1e-4$ for U-Net tuning. SGD method is not used in original repository. We emprically observe that SGD requires larger learning rate compared to AdamW to generate sensible images. For this experiment, we test with three groups of learning rates, with "Large" corresponds to $3e-1$ for SGD-type methods and $5e-4$ for

[2] https://civitai.com/models/6424/chilloutmix

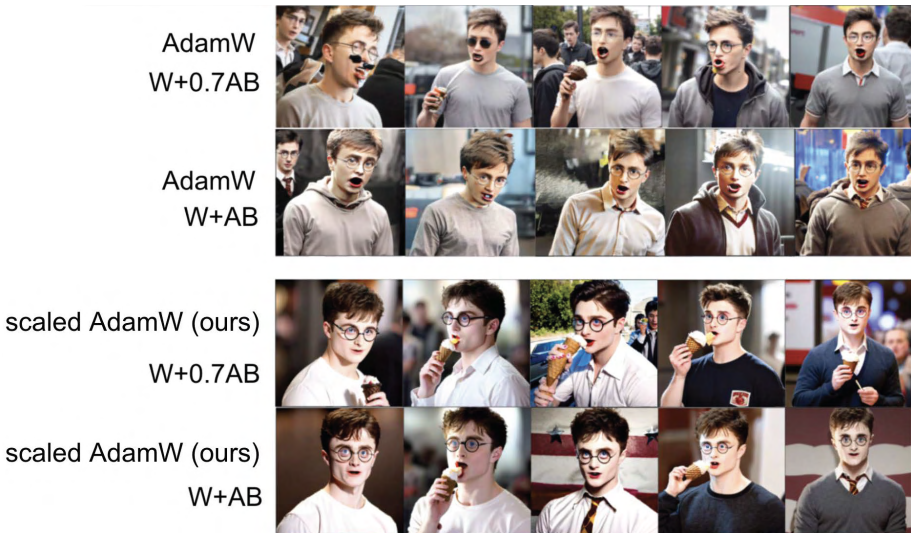

Figure 8. Generation results for prompt "$\langle V_{\text{potter}} \rangle$ eating an icecream " after fine-tuning on 14 Potter images. The above two rows are from AdamW optimizer and the bottom two rows are from our scaled AdamW optimizer. The first and third rows correspond to LoRA parameter fusion coefficient 0.7 and the second and fourth rows correspond to LoRA parameter fusion coefficient 1.0. Our scaled AdamW method generates images of higher quality compared to AdamW. See Appendix F.2.1 for experimental details.

Figure 9. Generation results for prompt "$\langle V_{\text{potter}} \rangle$" after fine-tuning on 14 Potter images. The above two rows are from AdamW optimizer and the bottom two rows are from our scaled AdamW optimizer. The first and third rows correspond to LoRA parameter fusion coefficient 0.7 and the second and fourth rows correspond to LoRA parameter fusion coefficient 1.0. Our scaled AdamW method generates images of higher quality compared to AdamW. See Appendix F.2.1 for experimental details.

AdamW-type methods; "Medium" corresponds to $1e-1$ for SGD-type methods and $1e-4$ for AdamW-type methods; "Small" corresponds to $5e-2$ for SGD-type methods and $1e-5$ for AdamW-type methods. We don't differentiate learning rates for U-Net and text-encoder tuning and the same learning rate is used for both. Note the default learning rate for AdamW falls between the "Medium" learning rate and the "Small" learning rate thus our learning rate choices are not random. Figure 12, 13 and 14 show the generation results for three different prompts. It can be observed that our scaled optimizers are more robust to learning rate changes.

Figure 10. Generation results for prompt "$\langle V_{\text{hermione}} \rangle$ wearing a red hat" after fine-tuning on 15 Hermione images. The above two rows are from AdamW optimizer and the bottom two rows are from our scaled AdamW optimizer. The first and third rows correspond to LoRA parameter fusion coefficient 0.7 and the second and fourth rows correspond to LoRA parameter fusion coefficient 1.0. Our scaled AdamW method generates images of higher quality compared to AdamW. See Appendix F.2.1 for experimental details.

Figure 11. Generation results for prompt "$\langle V_{\text{hermione}} \rangle$" after fine-tuning on 15 Hermione images. The above two rows are from AdamW optimizer and the bottom two rows are from our scaled AdamW optimizer. The first and third rows correspond to LoRA parameter fusion coefficient 0.7 and the second and fourth rows correspond to LoRA parameter fusion coefficient 1.0. Our scaled AdamW method generates images of higher quality compared to AdamW. See Appendix F.2.1 for experimental details.

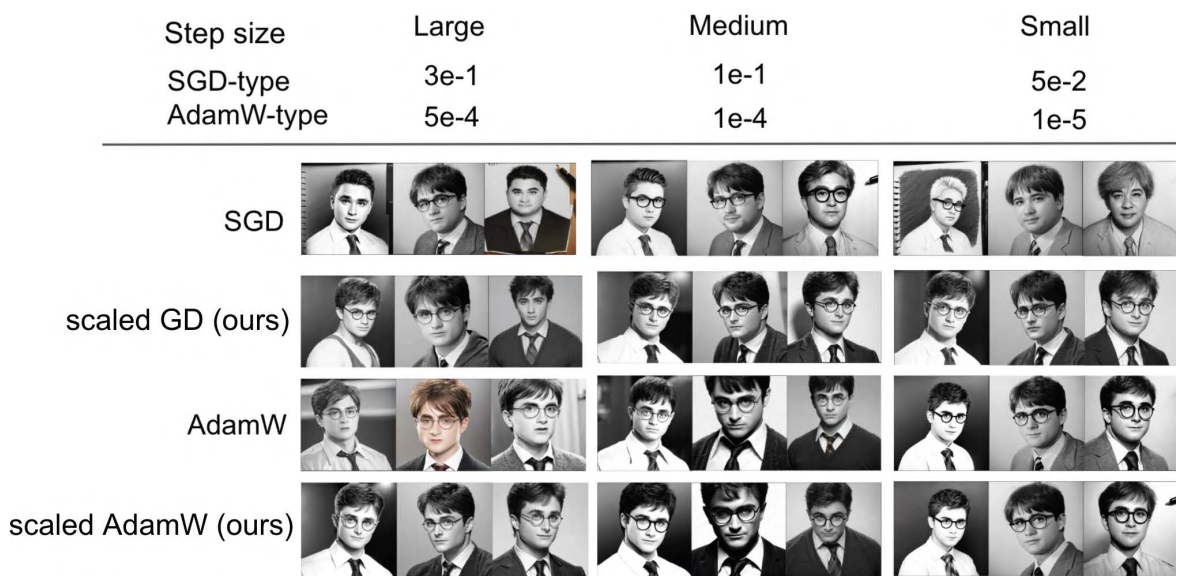

Figure 12. Generation results for prompt "a pencil sketch of $\langle V_{\text{potter}} \rangle$" with different optimizers and different learning rates. See Appendix F.2.2 for experimental details. Default optimizer is AdamW with default learning rate $1e-4$ for U-Net tuning and $1e-5$ for text-encoder tuning. Our scaled optimizers generate better quality images and are more robust to learning rate changes.

Figure 13. Generation results for prompt "$\langle V_{\text{potter}} \rangle$ sit on the chair" with different optimizers and different learning rates. See Appendix F.2.2 for experimental details. Default optimizer is AdamW with default learning rate $1e-4$ for U-Net tuning and $1e-5$ for text-encoder tuning. Our scaled optimizers generate better quality images and are more robust to learning rate changes.

Figure 14. Generation results for prompt "a photo of $\langle V_{\text{potter}} \rangle$" with different optimizers and different learning rates. See Appendix F.2.2 for experimental details. Default optimizer is AdamW with default learning rate $1e-4$ for U-Net tuning and $1e-5$ for text-encoder tuning. Our scaled optimizers generate better quality images and are more robust to learning rate changes.