AltLoRA: Towards Better Gradient Approximation in Low-Rank Adaptation with Alternating Projections

Xin Yu †
Department of Statistics
The Pennsylvania State University
State College, PA 16803
xmhy5152@psu.edu

Yujia Wang
College of Information Sciences and Technology
The Pennsylvania State University
State College, PA 16803
yjw5427@psu.edu

Jinghui Chen
College of Information Sciences and Technology
The Pennsylvania State University
State College, PA 16803
jzc5917@psu.edu

Lingzhou Xue‡
Department of Statistics
The Pennsylvania State University
State College, PA 16803
lzxue@psu.edu

Abstract

Low-Rank Adaptation (LoRA) has emerged as an effective technique for reducing memory overhead in fine-tuning large language models. However, it often suffers from sub-optimal performance compared with full fine-tuning since the update is constrained in the low-rank space. Recent variants such as LoRA-Pro attempt to mitigate this by adjusting the gradients of the low-rank matrices to approximate the full gradient. However, LoRA-Pro's solution is not unique, and different solutions can lead to significantly varying performance in ablation studies. Besides, to incorporate momentum or adaptive optimization design, approaches like LoRA-Pro must first compute the equivalent gradient, causing a higher memory cost close to full fine-tuning. A key challenge remains in integrating momentum properly into the low-rank space with lower memory cost. In this work, we propose AltLoRA, an alternating projection method that avoids the difficulties in gradient approximation brought by the joint update design, meanwhile integrating momentum without higher memory complexity. Our theoretical analysis provides convergence guarantees and further shows that AltLoRA enables stable feature learning and robustness to transformation invariance. Extensive experiments across multiple tasks demonstrate that AltLoRA outperforms LoRA and its variants, narrowing the gap toward full fine-tuning while preserving superior memory efficiency.

1 Introduction

Low-Rank Adaptation (LoRA [25]) has emerged as a leading approach for parameter-efficient fine-tuning (PEFT)([24, 38, 35]) of large language models ([5, 51, 61, 40]). Building on prior work investigating the intrinsic dimensionality of neural networks ([2, 36]), LoRA assumes that fine-tuning updates can be effectively captured in a low-rank subspace. Specifically, for a pre-trained model with weight matrix $W_0 \in \mathbb{R}^{k\times d}$, LoRA reparameterizes the weight update ΔW via a low-rank

†First Author.
‡Correspondence to: Lingzhou Xue<lzxue@psu.edu>.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

 https://doi.org/10.52202/085713-1258

decomposition as $W_0 + \Delta W = W_0 + sBA$, where $B \in \mathbb{R}^{k\times r}$, $A \in \mathbb{R}^{r\times d}$ and $s = \frac{\alpha}{r}$ is a scaling factor. Here, $r \ll \min(k, d)$ is the rank of the update. Thanks to its substantial memory and computational savings [25], LoRA has enabled scalable adaptation across diverse applications, including reinforcement learning from human feedback (RLHF) [57, 23], diffusion models [43, 77], and mixture-of-experts (MoE) architectures [67, 37].

Despite its parameter efficiency, LoRA often underperforms full fine-tuning ([13, 25, 41, 71]). This gap has fueled growing interest in optimizing LoRA via hyperparameter tuning under stable feature learning [21, 20] and optimizers that preserve transformation invariance [79]. Formally, if we denote the loss function as L, full fine-tuning will utilize the full gradient $\nabla_W L \in \mathbb{R}^{k\times d}$ for backpropagation. In contrast, the gradients in LoRA for B and A are given by $(\nabla_W L)A^\top$ and $B^\top(\nabla_W L)$, respectively (see Section 2). This reparameterization significantly alters the gradient flow during training [88] by restricting it to the low-rank space.

A promising direction to fill the gap between the gradient dynamics is to ensure that the equivalent gradient established by LoRA approximates the full gradient ([66, 65, 50]). However, two key challenges in the gradient approximation for low-rank adaptation remain unaddressed. First, LoRA-Pro [66] depends on an auxiliary variable that impacts the performance significantly. Depending on the choice of this variable, the evaluation score varies from 31.74 to 57.57 on the GSM8K datasets (see Appendix D.1 in [66]). Obtaining a unique solution requires solving a Sylvester equation, which introduces additional computational cost and relies on a non-standard assumption. Second, as LoRA-Pro accelerates the equivalent gradient with full-parameter learning, it requires a memory cost like full fine-tuning with space complexity $\mathcal{O}(kd)$ as shown in Table 1. In contrast, LoRA maintains a more efficient space complexity of $\mathcal{O}(kr + rd)$. Under such memory constraints, how to incorporate momentum properly within the low-rank structure is largely unexplored.

In this paper, to close the performance gap between LoRA and full fine-tuning, we address the two key challenges outlined above and propose a novel PEFT method, AltLoRA, based on **Alt**ernating updates to the **Lo**w-**R**ank **A**daptation. AltLoRA properly approximates the full gradient by alternately projecting it onto low-rank subspaces and B. Building on this projection-based gradient approximation, we further introduce a new mechanism to optimize momentum effectively within the low-rank space, while strictly adhering to the memory constraints of LoRA [25]. Without allowing full-parameter learning, AltLoRA is the first work in the literature to properly optimize both gradient and momentum over the low-rank subspaces, while achieving stable feature learning and transformation invariance, as summarized in Table 1.

Table 1: Comparison with Existing Work

Methods	Gradient Approximation	Stable Feature Learning	Transformation Invariance	Time Complexity	Space Complexity
LoRA [25]	✘	✘	✘	$\mathcal{O}(kr^2 + dr^2)$	$\mathcal{O}(kr + dr)$
LoRA+ [21]	✘	✔	✘	$\mathcal{O}(kr^2 + dr^2)$	$\mathcal{O}(kr + dr)$
ScaledAdam [81]	✘	✔	✘	$\mathcal{O}(kr^2 + dr^2)$	$\mathcal{O}(kr + dr)$
LoRA-Rite [79]	✘	✔	✔	$\mathcal{O}(kr^2 + dr^2)$	$\mathcal{O}(kr + dr)$
LoRA-Pro [66]	✔	✔	✔	$\mathcal{O}(kdr)$	$\mathcal{O}(kd)$
AltLoRA	✔	✔	✔	$\mathcal{O}(kr^2 + dr^2)$	$\mathcal{O}(kr + dr)$

Our main contributions are summarized as follows:

- We propose **AltLoRA**, a novel PEFT method that efficiently approximates the full gradient via alternating projections onto the low-rank subspaces A and B. Moreover, we design a new momentum mechanism that operates within LoRA's memory constraints, enabling effective optimization of momentum within the low-rank space.

- Theoretically, we prove that AltLoRA ensures stable feature learning in the infinite-width neural network regime and, more generally, maintains transformation invariance, even when incorporating momentum. We also provide convergence guarantees for fine-tuning overparameterized two-layer ReLU networks.

- Empirically, we show the effectiveness of AltLoRA through extensive experiments on tasks including natural language understanding, dialogue generation, mathematical reasoning, and code generation. AltLoRA consistently outperforms existing LoRA-based methods.

2 Preliminary

Let us first revisit the optimization paradigm of LoRA [25]. If we denote the loss function as L, i.e., $L(A, B) := L(W + sBA)$, we can derive the gradient w.r.t A and B as follows:

$$\nabla_A L := \frac{\partial L}{\partial A} = \frac{\partial L}{\partial W}\frac{\partial W}{\partial A} = sB^T(\nabla_W L), \quad \nabla_B L := \frac{\partial L}{\partial B} = \frac{\partial L}{\partial W}\frac{\partial W}{\partial B} = s(\nabla_W L)A^T. \tag{1}$$

Here, as the full gradient is multiplied by the low-rank matrices to constitute the gradient of LoRA, it implicitly compresses the full gradient into the low-rank spaces. Suppose we use gradient descent to update A and B, then the model parameter in the $(t+1)$-th iteration is:

$$\begin{aligned} W_{t+1} &= W_0 + sB_{t+1}A_{t+1} \\ &\approx W_0 + sB_tA_t - s\eta(\nabla_{B_t}L)A_t - s\eta B_t(\nabla_{A_t}L) \\ &= W_t - s\eta(\nabla_{B_t}L)A_t - s\eta B_t(\nabla_{A_t}L). \end{aligned} \tag{2}$$

Here, we omit the term related to η^2. Compared with the full gradient update $-\eta\nabla_W L$, LoRA's gradient can approximate the full gradient as long as $sB(\nabla_A L) + s(\nabla_B L)A$ is close to $\nabla_W L$. With a similar motivation, some previous work analyzes the approximation based on the Frobenius norm ([65, 66, 50]). Noticeably, LoRA-Pro [66] achieves gradient approxiation by adjusting the gradients of matrices A and B based on the following solutions:

$$g^A = \frac{1}{s}(B^TB)^{-1}B^T(\nabla_W L) + XA, g^B = \frac{1}{s}[I - B(B^TB)^{-1}B^T](\nabla_W L)A^T(AA^T)^{-1} - BX, \tag{3}$$

where $X \in \mathbb{R}^{r\times r}$ denotes an ancillary matrix and its selection is crucial and challenging for LoRA-Pro. As shown in their ablation studies, the selection of X would vary the performance of the evaluation significantly. Besides, to obtain a unique solution for X, LoRA-Pro imposes additional uncommon assumptions to solve a Sylvester equation. However, even selecting a unique X, the equivalent gradient($sBg^A + sg^BA$) established by LoRA-Pro is independent of X, which implies that X is only used to distinguish the gradient of A and B when jointly updating and doesn't influence the model update. It motivates the development of a more efficient alternating and eliminates the influence of X. To circumvent the ambiguity and inefficiency introduced by this joint updating strategy, we propose an alternating update strategy that approximates the full gradient as long as $sB(\nabla_A L)$ or $s(\nabla_B L)A$ is close to $\nabla_W L$.

Notation. Hereafter, we use the following notation to describe the asymptotic behavior as the width n grows. Given sequences $c_n \in \mathbb{R}$ and $d_n \in \mathbb{R}^+$, we write $c_n = \mathcal{O}(d_n)$, resp. $c_n = \Omega(d_n)$, to refer to $c_n < \kappa d_n$, resp. $c_n > \kappa d_n$, for some constant $\kappa > 0$. For vector and matrix sequences, the notation is applied entry-wise. Additionally, we use $\odot$ and $\oslash$ to denote element-wise matrix multiplication and division, respectively. $[P]$ denotes the set of indices $\{1, \cdots, P\}$.

3 Methodology

3.1 Alternately Approximating the Full Gradient via Low-Rank Adaptation

We propose an alternating update scheme, where we update A first and then update B based on the new A. Define the low-rank modules as A_t and B_t at the t-th iteration, and the approximated gradients as $\tilde{\nabla}_A L$ and $\tilde{\nabla}_B L$, respectively. We begin by obtaining the optimal scaling gradient of A by solving

$$\min_{\tilde{\nabla}_{A_t}L} \|sB_t(\tilde{\nabla}_{A_t}L) - \nabla_{W_t}L\|_F^2, \tag{4}$$

where $\|\cdot\|_F^2$ denotes the Frobenius norm squared—sum of squares of all entries in the matrix. Then by gradient descent, we can update A and the full model as

$$A_{t+1} \leftarrow A_t - \eta\tilde{\nabla}_{A_t}L, \quad W_{t+\frac{1}{2}} \leftarrow W_t - \eta B_t(\tilde{\nabla}_{A_t}L), \tag{5}$$

where we update the full model at $(t+1/2)$-th iteration to keep consistent with the joint update [66] (update A and B in one iteration). In our experiment, without any ambiguity, we treat the update A or

 https://doi.org/10.52202/085713-1258

B as a single step (see Algorithm 1). After doing backpropagation w.r.t A, the gradient of B doesn't approximate the full gradient at time t since the full model has been update to the state of $(t+1/2)$. Then we minimize the discrepancy between the full gradient at $W_{t+\frac{1}{2}}$ and the approximating gradient constructed by B_t as follow

$$\min_{\tilde{\nabla}_{B_t} L} \|s(\tilde{\nabla}_{B_t} L)A_{t+1} - \nabla_{W_{t+\frac{1}{2}}} L\|_F^2. \tag{6}$$

Then by gradient descent, we can update B and the full model as

$$B_{t+1} \leftarrow B_t - \eta\tilde{\nabla}_{B_t} L, \quad W_{t+1} \leftarrow W_{t+\frac{1}{2}} - \eta(\tilde{\nabla}_{B_t} L)A_{t+1}. \tag{7}$$

The following theorem gives the closed-form solution of Problems (4) and (6).

Theorem 1. *Assume $B_t \in \mathbb{R}^{k\times r}$ and $A_t \in \mathbb{R}^{r\times d}$ are full rank for any t, i.e. rank(B_t) = rank(A_t) = r. Solving Problems (4) and (6) yields the unique closed-form solutions*

$$\begin{aligned}
\tilde{\nabla}_{A_t} L &= \frac{1}{s}(B_t^T B_t)^{-1} B_t^T (\nabla_{W_t} L) = \frac{1}{s^2}(B_t^T B_t)^{-1}\nabla_{A_t} L \\
\tilde{\nabla}_{B_t} L &= \frac{1}{s}\left(\nabla_{W_{t+\frac{1}{2}}} L\right) A_{t+1}^T (A_{t+1}A_{t+1}^T)^{-1} = \frac{1}{s^2}\nabla_{B_t} L(A_{t+1}A_{t+1}^T)^{-1},
\end{aligned} \tag{8}$$

where $\nabla_{A_t} L$ and $\nabla_{B_t} L$ are the gradients of LoRA defined in Equation (1).

Theorem 1 shows that both problems admit unique optimal solutions for $\tilde{\nabla}_{A_t} L$ and $\tilde{\nabla}_{B_t} L$, which only requires full rank. Therefore, it offers a new gradient approximation with less computational cost and promotes a more efficient updating strategy. Besides, instead of accessing the full gradient like full fine-tuning, the optimal gradient approximation only requires the standard gradient of A or B by backpropagation at each step and calculating the inverse of a small matrix with size $r \times r$.

Theorem 1 requires that the matrix B_t and A_t are full rank, but in the over-parameterized cases, the assumption is hard to achieve. To alleviate it, if we penalize the Frobenius norm of these two approximated gradients, i.e., weight decay, the condition can be eliminated (see Corollary 1). For simplicity, in the rest of the paper, we focus on the modified gradient in (8) for analysis. The closed-form solution in (8) yields the following full model update(with gradient descent)

$$\begin{aligned}
W_{t+1} &= W_{t+\frac{1}{2}} - \eta(\tilde{\nabla}_{B_t} L)A_{t+1} \\
&= W_{t+\frac{1}{2}} - \eta(\nabla_{W_{t+\frac{1}{2}}} L)A_{t+1}^T(A_{t+1}A_{t+1}^T)^{-1}A_{t+1} \\
&= W_t - \eta B_t\tilde{\nabla}_{A_t} L - \eta(\nabla_{W_{t+\frac{1}{2}}} L)A_{t+1}^T(A_{t+1}A_{t+1}^T)^{-1}A_{t+1} \\
&= W_t - \eta B_t(B_t^T B_t)^{-1}B_t^T(\nabla_{W_t} L) - \eta(\nabla_{W_{t+\frac{1}{2}}} L)A_{t+1}^T(A_{t+1}A_{t+1}^T)^{-1}A_{t+1} \\
&= W_t - \eta Proj_{c(B_t)}(\nabla_{W_t} L) - \eta(\nabla_{W_{t+\frac{1}{2}}} L)Proj_{r(A_{t+1})}.
\end{aligned} \tag{9}$$

Interestingly, the proposed solution for gradient approximation in (8), is consistent with the literature work [59, 83, 73, 30, 42] called scaled gradient descent [46, 45] in low-rank matrix estimation [54]. Therefore, the view of gradient approximation would provide a novel interpretation of applying scaled gradient descent within the broader context of low-rank matrix decomposition. As optimizing LoRA with momentum for acceleration is a standard way in the literature [8, 25, 21], we will discuss how to properly design momentum within the low-rank space inspired by gradient approximation.

3.2 Proper Momentum Design within the Low-Rank Subspaces

For LoRA [25] and its variants [21, 86] without allowing full-parameter learning, the parameterization restricts both the gradient and the momentum updates to low-rank subspaces as the memory cost is $\mathcal{O}(kr + dr)$. As we have shown, the optimal gradient approximation under this constraint is obtained by projecting the full gradient onto the low-rank subspace. This insight naturally motivates the need to also align the momentum optimally within the same low-rank space, in order to fully leverage momentum-based acceleration under low-rank constraints.

Since the momentum evolves throughout training, it is essential to dynamically optimize it. For simplicity, we focus on the optimization paradigm for B and develop our method inductively. Given

the aligned momentum M_t^B within the low-rank space A_t at time t, the alternating update strategy proceeds by updating A to A_{t+1} and then aligning M_t^B with the new low-rank space A_{t+1}. To this end, we first recover M_t^B to the full-dimensional space, and then project it onto the new subspace spanned by A_{t+1}, like gradient approximation. The following theorem formalizes this key idea.

Theorem 2. *Assume $A_{t+1}A_{t+1}^T$ is full-rank, i.e., rank$(A_{t+1}A_{t+1}^T) = r$. If M_t^B has aligned with the low-rank space A_t in the t-th iteration, by minimizing the following problem*

$$\min_{\tilde{M}_t^B} \|M_t^B A_t - \tilde{M}_t^B A_{t+1}\|_F^2. \tag{10}$$

We can find $\tilde{M}_t^B = M_t^B A_t A_{t+1}^T (A_{t+1}A_{t+1}^T)^{-1}$, which makes the momentum aligned with the new low-rank space A_{t+1} optimally.

Theorem 2 shows that it is only necessary to store two small matrices so that we can optimize momentum properly. Similarly to Section 3.1, we can also remove the assumption of full rank here (see Corollary 2). In contrast to LoRA-Pro with full-parameter learning (Space Complexity $\mathcal{O}(kd)$), we aim to strictly satisfy the space complexity $\mathcal{O}(kr + dr)$ for parameter efficiency and keep momentum adaptively aligned with the low-rank spaces as the gradient approximation does.

A similar notion of momentum design is explored in [18, 22], where down-projection and up-projection matrices are employed to transfer compressed gradients across low-rank spaces. In contrast, we derive the optimal alignment directly within the low-rank subspaces to preserve gradient information. In Section 4.2, we theoretically demonstrate that aligning momentum with low-rank space guaranties formation invariance, whereas LoRA [25] and its variants [21, 86] have misaligned momentum that undermines this robustness [79].

After analyzing how to efficiently optimize both the gradient and momentum under limited resource constraints, we summarize our proposed algorithm, AltLoRA, in Algorithm 1. Unlike the joint update strategy, AltLoRA updates only one of the low-rank matrices, either A or B, at each step, based on the scaled gradient and momentum presented in Theorems 1 and 2. The number of trainable parameters at each step is reduced by half compared to the joint update. Designed as a practical PEFT method, AltLoRA can be seamlessly integrated into existing libraries such as Hugging Face [69] (see Appendix C.1 for implementation details). To further accelerate and stabilize the training paradigm of AltLoRA, we introduce AltLoRA+, an enhanced variant that naturally incorporates second-moment estimates similar to AdamW (see Algorithm 2 for details).

Algorithm 1: AltLoRA: Gradient Approximation via Alternating Projection with Proper Momentum Design under LoRA's Memory Constraint

Input: Momentum states M_0^A, M_0^B; scaling factor $s = \frac{\alpha}{r}$; learning rate η; momentum coefficient β_1; total steps T; weight decay γ

Output: Final matrices A_T and B_T

for $t = 0, \ldots, T-1$ **do**

  if $t \bmod 2 = 0$ **then**

    Update A**:**

      Only backpropagate w.r.t. A_t and obtain $\nabla_{A_t} L$

      $\tilde{\nabla}_{A_t} L = \frac{1}{s^2}(B_t^\top B_t)^{-1} \nabla_{A_t} L$

      $\tilde{M}_t^A = (B_t^\top B_t)^{-1} B_t^T B_{t-1} M_{t-1}^A$

      $M_t^A \leftarrow \beta_1 \tilde{M}_t^A + (1-\beta_1)\tilde{\nabla}_{A_t} L$

      $A_{t+1} \leftarrow A_t - \eta(M_t^A + \gamma A_t)$

  else

    Update B**:**

      Only backpropagate w.r.t. B_t and obtain $\nabla_{B_t} L$

      $\tilde{\nabla}_{B_t} L = \frac{1}{s^2} \nabla_{B_t} L (A_{t+1}A_{t+1}^\top)^{-1}$

      $\tilde{M}_t^B = M_{t-1}^B A_t A_{t+1}^\top (A_{t+1}A_{t+1}^\top)^{-1}$

      $M_t^B \leftarrow \beta_1 \tilde{M}_t^B + (1-\beta_1)\tilde{\nabla}_{B_t} L$

      $B_{t+1} \leftarrow B_t - \eta(M_t^B + \gamma B_t)$

Time Complexity and Space Complexity. When $r \ll min\{k, d\}$, the time and memory cost of AltLoRA and AltLoRA+ is similar to the standard LoRA and more efficient compared with LoRA-

https://doi.org/10.52202/085713-1258

Pro. The additional computational cost takes $\mathcal{O}(r^3)$ time, and since r is very small, this overhead is negligible when compared with the back-propagating time. In the experiment, we will show that the delay time compared with LoRA is mild even when the rank r increases. (see Table 3).

4 Theoretical Analysis

4.1 Stable Feature Learning

Given the current trend of increasing model sizes ([76, 47, 75]), it raises a lot of attention to analyze the asymptotic training behavior of neural networks as the number of neurons approaches infinity ([56, 19, 74]). There is a line of work in LoRA ([21, 20, 81]) considering the infinite-width NN setting. To achieve stable feature learning (see Definition 2 in Appendix D.1), they propose a fine-grained choice of hyperparameters in the original LoRA, like the learning rate [21], the initialization ([20]), and the optimizer ([81]). The core idea is that the update increment over the loss function or parameter should 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. First, we demonstrate that our method achieves stable feature learning on a toy model in Appendix D.1.1. We then prove that this stability extends to arbitrary LoRA ranks and holds for AltLoRA and AltLoRA+, which we formalize in the theorem below. For clarity of presentation, we omit the scaling factor s in the subsequent theorems and analysis.

Theorem 3 (Informal)**.** *Assume that, with the input x, BAx has dimension $\mathcal{O}(n)$. In Algorithm 1 or Algorithm 2, if we use the same learning rate $\eta = \mathcal{O}(1)$ to update A and B, it would achieves stable feature learning. Moreover, without momentum in AltLoRA or AltLoRA+, the model update achieves stable feature learning as well with*

$$W_{t+1} = W_t - \eta Proj_{c(B_t)}(\nabla_{W_t} L) - \eta(\nabla_{W_{t+\frac{1}{2}}} L) Proj_{r(A_{t+1})}, \tag{11}$$

where

$$\eta Proj_{c(B_t)}(\nabla_{W_t} L), \eta(\nabla_{W_{t+\frac{1}{2}}} L) Proj_{r(A_{t+1})} \in \mathcal{O}(1).$$

However, when doing joint update ([81]), the update will introduce additional across term $\eta^2(\nabla_{W_t} L)A_t^T(A_t A_t^T)^{-1}(B_t^T B_t)^{-1} B_t^T(\nabla_{W_t} L) \in \mathcal{O}(1)$. The across term is indeed the second order term w.r.t η, but it is same magnitude as $\eta(\nabla_{W_t} L)Proj_{r(A_t)}$ and $\eta Proj_{c(B_t)}(\nabla_{W_t} L)$ in infinite-width NN setting.

In Theorem 3, AltLoRA and AltLoRA+ achieve stable feature learning. Moreover, as the joint update would introduce the cross term with an unignorable magnitude (especially η is $\mathcal{O}(1)$ instead of $\mathcal{O}(1/n)$ in the toy model), joint update with scaled gradient descent ([81]) breaks the clean interpretation of projecting the full gradient onto low-rank subspaces and degrade the performance as our experiment studies show later.

4.2 Transformation Invariance

With the motivation that an optimizer should yield the same update to the full model regardless of the specific factorization, transformation invariance, as a sufficient condition for stable feature learning, is proposed by LoRA-RITE [79]. Here, we will prove that our designed gradient and momentum in Algorithm 1 would be inherently robust as transformation invariance.

Definition 1. *If there are two pairs of LoRA matrix $(A_1, B_1), (A_2, B_2)$ can represent the same finetuned weight $W = W_0 + B_1 A_1 = W_0 + B_2 A_2$. An optimizer exhibits transformation invariance if its updates, $(\delta A_1, \delta B_1)$ and $(\delta A_2, \delta B_2)$ satisfy*

$$\begin{aligned} &W_0 + (B_1 + \delta B_1)(A_1 + \delta A_1) = W_0 + (B_2 + \delta B_2)(A_2 + \delta A_2) \\ &\Rightarrow (B_1 + \delta B_1)(A_1 + \delta A_1) = (B_2 + \delta B_2)(A_2 + \delta A_2). \end{aligned} \tag{12}$$

LoRA-RITE [79] notices that, after combining scaled gradient descent with element-wise Adam in [81], the ScaledAdam can't preserve transformation invariance. As the momentum is optimized properly, we will analyze how AltLoRA keeps transformation invariance naturally, especially when incorporating momentum.

Recall the definition of projection matrices in Equation (9): $Proj_{c(B_t)} := B_t(B_t^T B_t)^{-1} B_t^T$ (or $Proj_{r(A_t)} := A_t^T(A_t A_t^T)^{-1} A_t$). The following lemma provides insight into how Algorithm 1 achieves transformation invariance.

Lemma 1. *If any two pairs of LoRA factors* $(A_1, B_1), (A_2, B_2)$ *satisfying*

$$W = W_0 + B_1 A_1 = W_0 + B_2 A_2, \tag{13}$$

then $$Proj_{c(B_1)} = Proj_{c(B_2)}, \quad Proj_{r(A_1)} = Proj_{r(A_2)}\,.$$

Even though the full model update can be decomposed into different pairs of low-rank adaptations, within each pair of LoRA factors, the column space of B (or the row space of A) is equivalent to the column space (or the row space) of the full model update. Therefore, the projection matrix would be preserved invariant over the pairs of low-rank adaptation.

Theorem 4. *AltLoRA in Algorithm 1 is transformation-invariant.*

Building on the insight from Lemma 1, we leverage the invariance of the projection matrix to the low-rank subspaces to approximate the full gradient via the gradient and moment information. As a result, with the goal of gradient approximation without full-parameter learning, our method achieves transformation invariance inherently. LoRA-RITE [79] is also aware of the equivalence of low-rank spaces, but they do not notice or exploit the invariance of the projection matrix. Instead, they design an unmagnified gradient requiring polar decomposition at each iteration, which introduces additional computational overhead. In contrast, our method avoids polar decomposition, contributing to its superior efficiency (see Table 3). LoRA-Pro [66] also achieves transformation invariance but does so without adhering to LoRA's memory constraint. AltLoRA in Algorithm 1, by comparison, strictly follows the memory budget of LoRA while preserving transformation invariance through a more efficient design. While Algorithm 2 does not currently maintain transformation invariance under second-order momentum, this opens an exciting avenue for future research. In Appendix D.2, we provide a detailed discussion on why extending our first-order momentum design to the second order poses fundamental challenges. Despite this, AltLoRA+ achieves substantial empirical gains over LoRA and its variants, demonstrating the practical strength of our approach even when we only keep the transformation-invariant up to the second momentum.

4.3 Convergence Analysis

Following [81], we provide a convergence analysis of AltLoRA (or AltLoRA+) without momentum within the over-parameterized two-layer ReLU NN tuning problem (see Appendix D.3). In Theorem 7, we show that the convergence is independent of the condition number of the data matrix. In contrast to [81], we impose fewer assumptions to establish the convergence analysis. Notably, we don't require the extended spectral initialization in Definition 7.3 [81]. In our experimental study, AltLoRA (AltLoRA+) can achieve superior performance with the variant of initialization used by LoRA and its variants (see Appendix E.3.2), which supports our insight empirically.

5 Experimental Results

This section empirically shows the effectiveness of our approach across various model architectures and datasets. Section 5.1 summarizes the experimental settings and results on supervised fine-tuning (SFT) benchmark tasks, and Section 5.2 provides details of the setup and results for natural language understanding tasks. Finally, ablation studies from multiple perspectives are presented in Section 5.3. The code for our project is available at https://github.com/LucasXinYu/AltLoRA.

5.1 Experiments on SFT of LLM: Natural Language Generation

Training Details. We assess our methods on dialogue generation with the WizardLM dataset [72], mathematical reasoning with the MetaMathQA dataset [80], and code generation with the CodeFeedBack dataset [90] using the LLama-3.1-8B and Llama-3-8B models [17] (see Appedix E.1). We compare AltLoRA and AltLoRA+ with the pretrained model, full fine-tuning, LoRA [25], PisSSA[44], rsLoRA[31], LoRA+[21], DoRA[41], AdaLoRA[86], LoRA-GA[65], LoRA-Rite [79]and LoRA-Pro[66]. To ensure fair comparisons, we closely follow the experimental protocol established by [66]. Unless otherwise stated, we fine-tune models using default hyperparameters (if

 https://doi.org/10.52202/085713-1258

used): $\beta_1 = 0.9$, $\beta_2 = 0.999$, and zero weight decay. We adopt a cosine learning rate schedule with a warm-up ratio of 0.03. LoRA adapters are applied to $\{Q, K, V, O\}$ layers. By default, we set the rank to $r = 8$ and the scaling factor to $\alpha = 32$ for dialogue generation tasks, and $r = 8$, $\alpha = 16$ for the mathematical reasoning and code generation tasks. We carefully grid search the learning rates [‡]. To obtain a reliable estimate of model performance, we perform three runs with different random seeds and report the average and standard deviation of the results.

Evaluations. We evaluate the baselines similar to [66]. Specifically, for the dialogue generation task, we use the MT-Bench dataset [89] with GPT-4o, with scores ranging from 1 to 10. We report the score from the first turn as our metric. For the math task, we evaluate the model on the GSM8K test set [11] using the LLM Evaluation Harness [16], and we report the exact match accuracy. For the code generation task, we evaluate on the HumanEval dataset [6] and report the PASS@1 metric.

Results. Table 2 presents our experimental results, which demonstrates AltLoRA superior performance. With a rank of 8, AltLoRA achieves noticeable improvement over the original LoRA: 0.5 on MT-bench, 8.38 on GSM8K and 3.1 on HumanEval using Llama-3.1-8B. Notably, AltLoRA achieves significantly higher scores on MT-Bench compared to LoRA-Pro and Full FT. In addition, AltLoRA+ yields improvements over LoRA-Pro on both GSM8K and HumanEval, and AltLoRA+ obtains better performance in mathematical reasoning than Full FT. These further demonstrate the effectiveness of the new design gradient and momentum. The additional study on Llama-3-8B model (see Table 5 in Appendix E.1) also demonstrates a clear advantage over baseline methods.

Table 2: Comparison of different LoRA variants on MT-Bench, GSM8K, and HumanEval benchmarks on Llama-3.1-8B-Base. **Bold** indicates the best result, underline represents the second-best one.

Method	MT-Bench	GSM8K	HumanEval
PreTrain	5.93±0.08	51.34±1.38	36.15±1.97
Full FT	6.31±0.04	73.31±0.32	**50.81±1.10**
LoRA	6.06±0.02	66.11±1.43	40.31±1.34
PiSSA	5.15±0.10	67.78±1.11	42.44±1.11
rsLoRA	6.10±0.06	68.12±0.44	43.91±1.44
LoRA+	6.40±0.06	72.33±1.33	44.10±1.38
DoRA	6.08±0.03	68.33±0.88	42.13±1.31
AdaLoRA	6.08±0.05	72.63±1.45	42.21±2.66
LoRA-GA	6.00±0.09	70.33±0.91	42.01±1.21
LoRA-Pro	6.19±0.03	73.12±0.56	43.13±1.45
LoRA-Rite	6.10±0.01	74.10±0.31	43.12±0.51
AltLoRA	**6.56±0.04**	74.49±0.57	45.91±1.14
AltLoRA (rank=32)	6.39±0.04	73.24±0.29	46.87±1.49
AltLoRA (rank=128)	6.27±0.01	74.11±0.21	45.41±1.65
AltLoRA+	6.16±0.02	**76.91±0.31**	50.10±1.35
AltLoRA+ (rank=32)	6.10 ±0.02	76.32±0.29	49.97±1.52
AltLoRA+ (rank=128)	6.07±0.03	77.08±0.83	49.77±1.58

Memory and Time Consumptions. In Table 3, we also compare the memory cost and training time of our methods with Full FT, LoRA, LoRA-Rite and LoRA-Pro on Llama-3.1-8b mode. Without full-parameter learning, we have a comparable memory cost and training time close to LoRA. After taking a higher rank of LoRA, the memory cost and computation cost won't increase significantly. However, as LoRA-Pro requires storing the full size first-order momentum and second-order momentum, it leads to an unignorable cost like Full FT. As LoRA-Rite incurs additional calculations like polar decomposition, it also increase the computation time.

[‡]See Appendix E.1 for details of learning rate grid search. We set the sequence length to 1024 and the macro batch size to 4 for math and code tasks, and macro batch size to 8 for dialogue generation. All experiments are conducted on NVIDIA A100 and NVIDIA A6000 GPUs.

Table 3: Comparison of memory usage and training time across different fine-tuning methods.

Method	**Memory Cost**	**Training Time**
Full FT	$>$ 48 GB	4h 23min
LoRA	22.26 GB	2h 13min
LoRA-Rite	25.39 GB	2h 44min
LoRA-Pro	40.12 GB	4h 5min
AltLoRA	22.56 GB	2h 34min
AltLoRA(rank=32)	23.11 GB	2h 41min
AltLoRA(rank=128)	25.11 GB	2h 52min
AltLoRA+	23.16 GB	2h 38min
AltLoRA+(rank=32)	24.98 GB	2h 45min
AltLoRA+(rank=128)	27.76 GB	2h 56min

5.2 Experiments on Natural Language Understanding

Training and Evaluation Details. We assess our methods natural language understanding on a subset of GLUE benchmark dataset with fine-tuning a T5-base[52] model. We compare AltLoRA and AltLoRA+ with the full fine-tuning, LoRA [25], PisSSA[44], rsLoRA[31], LoRA+[21], DoRA[41], AdaLoRA[86], LoRA-GA[65], and LoRA-Pro[66]. We fine-tune the T5-based model [52] with our methods and the baselines on a subset of GLUE datasets [63]: MNLI, SST2, CoLA, QNLI, and MRPC. We use the accuracy as the evaluation metric. To ensure fair comparison, all experiments are run three times with different random seeds, and we report the mean and standard deviation of the results. Due to space constraints, additional experimental details are provided in Appendix E.1.

Results. As shown in Table 4, AltLoRA+ outperforms the baselines on average. In particular, it achieves the highest score on MRPC, the second-highest on CoLA, MNLI, and SST-2 datasets.

Table 4: Performance of fine-tuning T5-Base on 5 sub-tasks of the GLUE benchmark. **Bold** indicates the best result, underline represents the second-best one, and * marks results reported from [65].

Method	**MNLI**	**SST-2**	**CoLA**	**QNLI**	**MRPC**	**Average**
Full	**86.29±0.01**	93.97±0.06	80.87±0.05	93.02±0.03	86.89±0.13	88.21
LoRA	85.32±0.01	93.76±0.05	81.31±0.20	92.96±0.09	86.03±0.24	87.88
RSLoRA	85.23±0.01	93.96±0.06	81.21±0.14	93.12±0.09	86.27±0.24	87.96
DoRA	85.58±0.03	93.65±0.06	81.16±0.04	93.04±0.06	86.14±0.12	87.91
LoRA+	85.32±0.06	93.92±0.11	81.21±0.06	92.97±0.03	86.25±0.16	87.93
PiSSA	85.87±0.04	93.84±0.06	**81.90±0.05**	93.16±0.09	86.64±0.12	88.28
LoRA-GA*	85.70±0.09	**94.11±0.18**	80.57±0.20	**93.18±0.06**	85.29±0.24	87.77
AdaLoRA	85.45±0.11	93.92±0.09	80.31±0.05	91.66±0.05	86.16±0.60	87.50
LoRA-Pro	85.70±0.11	93.92±0.10	78.42±0.03	93.15±0.03	86.54±0.50	87.55
AltLoRA	85.26±0.04	93.87±0.05	80.44±0.09	91.56±0.01	86.60±0.99	87.55
AltLoRA+	85.81±0.03	94.03±0.12	81.44±0.30	92.99±0.03	**87.25±1.12**	**88.30**

5.3 Ablation Study

Figure 1 presents an ablation study of the learning rate η and the scaling factor α for LoRA, AltLoRA and AltLoRA+, using the LLaMA 3.1-8B model on mathematical reasoning tasks. The results show that our proposed methods are robust in learning rate and the scaling factor with consistent superior performance. Moreover, it shows that $\alpha = 16$ obtains overall better performance compared to $\alpha = 8$ and $\alpha = 32$. The influence of increasing rank is reported in Table 2 (see Appendix E.3 of the results on Llama-3-8B model). Besides, studying the choice of hyperparameters, in Appendix E.3.2, we present additional ablation studies on the Llama 3.1-8B model as well. To evaluate the effectiveness of alternating strategies, we compare them against the joint update method. As the approaches of multiple LoRA modules, such as in the mixture of LoRA experts, has gained popularity [37, 70], we also assess the impact of varying the number of experts in LoRA layers. Finally, to further validate the robustness of our method with respect to initialization, as discussed in Section 4.3, we study

 https://doi.org/10.52202/085713-1258

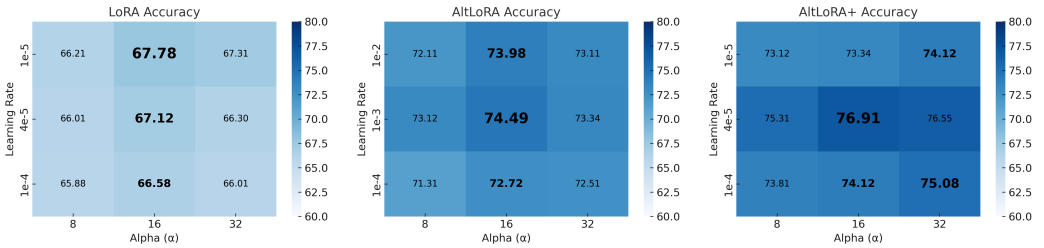

Figure 1: Evaluation Accuracy of LoRA, AltLoRA and AltLoRA+ for various learning rate η and scaling factor α combination on the GSM8K datasets using Llama-3.1-8B.

different initialization strategies. These ablation studies collectively demonstrate that our method is robust to hyperparameter variations and is applicable to more complex model architectures.

6 Conclusion

We propose AltLoRA, a memory-efficient fine-tuning method that alternates updates of low-rank matrices to dynamically project both the gradient and momentum within low-rank subspaces. By leveraging an efficient closed-form gradient approximation and a principled momentum design, AltLoRA operates entirely under low-rank constraints while ensuring stable feature learning and transformation invariance without requiring full-parameter learning. Extensive experiments across diverse tasks demonstrate the superior performance of AltLoRA and its enhanced variant, AltLoRA+, over LoRA and its variants, narrowing the gap to full fine-tuning while retaining memory efficiency.

Acknowledgements

The work of X. Yu and L. Xue was supported by the U.S. National Science Foundation under the grants CCF-2007823 and DMS-2210775, and by the U.S. National Institutes of Health under the grant 1R01GM152812. The work of Y. Wang and J. Chen was partially supported by the National Science Foundation under Grant No. 2348541. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

 https://doi.org/10.52202/085713-1258

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*, 2020.

[3] Jiamu Bai, Daoyuan Chen, Bingchen Qian, Liuyi Yao, and Yaliang Li. Federated fine-tuning of large language models under heterogeneous language tasks and client resources. *arXiv e-prints*, pages arXiv–2402, 2024.

[4] João Carlos Alves Barata and Mahir Saleh Hussein. The moore–penrose pseudoinverse: A tutorial review of the theory. *Brazilian Journal of Physics*, 42:146–165, 2012.

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

[6] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[7] Shuangyi Chen, Yuanxin Guo, Yue Ju, Harik Dalal, and Ashish Khisti. Robust federated finetuning of llms via alternating optimization of lora. *arXiv preprint arXiv:2502.01755*, 2025.

[8] Yiming Chen, Yuan Zhang, Liyuan Cao, Kun Yuan, and Zaiwen Wen. Enhancing zeroth-order fine-tuning for language models with low-rank structures. *arXiv preprint arXiv:2410.07698*, 2024.

[9] Yiming Chen, Yuan Zhang, Yin Liu, Kun Yuan, and Zaiwen Wen. A memory efficient randomized subspace optimization method for training large language models. *arXiv preprint arXiv:2502.07222*, 2025.

[10] Cheng Cheng and Ziping Zhao. Accelerating gradient descent for over-parameterized asymmetric low-rank matrix sensing via preconditioning. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7705–7709. IEEE, 2024.

[11] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[12] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

[13] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, 2023.

[14] Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, et al. Loramoe: Alleviate world knowledge forgetting in large language models via moe-style plugin. *arXiv preprint arXiv:2312.09979*, 2023.

[15] Ke-Lin Du, MNS Swamy, Zhang-Quan Wang, and Wai Ho Mow. Matrix factorization techniques in machine learning, signal processing, and statistics. *Mathematics*, 11(12):2674, 2023.

[16] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024. URL `https://zenodo.org/records/12608602`.

[17] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[18] Yongchang Hao, Yanshuai Cao, and Lili Mou. Flora: Low-rank adapters are secretly gradient compressors. *arXiv preprint arXiv:2402.03293*, 2024.

[19] Soufiane Hayou, Arnaud Doucet, and Judith Rousseau. On the impact of the activation function on deep neural networks training. In *International conference on machine learning*, pages 2672–2680. PMLR, 2019.

[20] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. The impact of initialization on lora finetuning dynamics. *Advances in Neural Information Processing Systems*, 37:117015–117040, 2024.

[21] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*, 2024.

[22] Yutong He, Pengrui Li, Yipeng Hu, Chuyan Chen, and Kun Yuan. Subspace optimization for large language models with convergence guarantees. *arXiv preprint arXiv:2410.11289*, 2024.

[23] Jessica Hoffmann, Christiane Ahlheim, Zac Yu, Aria Walfrand, Jarvis Jin, Marie Tano, Ahmad Beirami, Erin van Liemt, Nithum Thain, Hakim Sidahmed, et al. Improving neutral point of view text generation through parameter-efficient reinforcement learning and a small-scale high-quality dataset. *arXiv preprint arXiv:2503.03654*, 2025.

[24] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019.

[25] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1 (2):3, 2022.

[26] Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. Lorahub: Efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv:2307.13269*, 2023.

[27] Qiushi Huang, Tom Ko, Zhan Zhuang, Lilian Tang, and Yu Zhang. Hira: Parameter-efficient hadamard high-rank adaptation for large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[28] Nam Hyeon-Woo, Moon Ye-Bin, and Tae-Hyun Oh. Fedpara: Low-rank hadamard product for communication-efficient federated learning. *arXiv preprint arXiv:2108.06098*, 2021.

[29] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 665–674, 2013.

[30] Xixi Jia, Hailin Wang, Jiangjun Peng, Xiangchu Feng, and Deyu Meng. Preconditioning matters: Fast global convergence of non-convex matrix factorization via scaled gradient descent. *Advances in Neural Information Processing Systems*, 36:76202–76213, 2023.

[31] Damjan Kalajdzievski. A rank stabilization scaling factor for fine-tuning with lora. *arXiv preprint arXiv:2312.03732*, 2023.

[32] Siddhartha Rao Kamalakara, Acyr Locatelli, Bharat Venkitesh, Jimmy Ba, Yarin Gal, and Aidan N Gomez. Exploring low rank training of deep neural networks. *arXiv preprint arXiv:2209.13569*, 2022.

[33] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023.

 https://doi.org/10.52202/085713-1258

[34] Dawid J Kopiczko, Tijmen Blankevoort, and Yuki M Asano. Vera: Vector-based random matrix adaptation. *arXiv preprint arXiv:2310.11454*, 2023.

[35] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[36] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. *arXiv preprint arXiv:1804.08838*, 2018.

[37] Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan Zhang, Lei Duan, Jie Zuo, Cal Yang, et al. Mixlora: Enhancing large language models fine-tuning with lora-based mixture of experts. *arXiv preprint arXiv:2404.15159*, 2024.

[38] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[39] Xutao Liao, Shaohui Li, Yuhui Xu, Zhi Li, Yu Liu, and You He. Galore +: Boosting low-rank adaptation for llms with cross-head projection. *arXiv preprint arXiv:2412.19820*, 2024.

[40] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[41] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*, 2024.

[42] Zhiyu Liu, Zhi Han, Yandong Tang, Hai Zhang, Shaojie Tang, and Yao Wang. Efficient over-parameterized matrix sensing from noisy measurements via alternating preconditioned gradient descent. *arXiv preprint arXiv:2502.00463*, 2025.

[43] Simian Luo, Yiqin Tan, Suraj Patil, Daniel Gu, Patrick von Platen, Apolinário Passos, Longbo Huang, Jian Li, and Hang Zhao. Lcm-lora: A universal stable-diffusion acceleration module. *arXiv preprint arXiv:2311.05556*, 2023.

[44] Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. *Advances in Neural Information Processing Systems*, 37:121038–121072, 2024.

[45] Bamdev Mishra and Rodolphe Sepulchre. Riemannian preconditioning. *SIAM Journal on Optimization*, 26(1):635–660, 2016.

[46] Bamdev Mishra, K Adithya Apuroop, and Rodolphe Sepulchre. A riemannian geometry for low-rank matrix completion. *arXiv preprint arXiv:1211.1550*, 2012.

[47] Lorenzo Noci, Chuning Li, Mufan Li, Bobby He, Thomas Hofmann, Chris J Maddison, and Dan Roy. The shaped transformer: Attention models in the infinite depth-and-width limit. *Advances in Neural Information Processing Systems*, 36:54250–54281, 2023.

[48] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*, 2020.

[49] Mert Pilanci and Tolga Ergen. Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks. In *International Conference on Machine Learning*, pages 7695–7705. PMLR, 2020.

[50] Kaustubh Ponkshe, Raghav Singhal, Eduard Gorbunov, Alexey Tumanov, Samuel Horvath, and Praneeth Vepakomma. Initialization using update approximation is a silver bullet for extremely efficient low-rank fine-tuning. *arXiv preprint arXiv:2411.19557*, 2024.

[51] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[52] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[53] Benjamin Recht, Maryam Fazel, and Pablo A Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501, 2010.

[54] Angelika Rohde and Alexandre B Tsybakov. Estimation of high-dimensional low-rank matrices1. *The Annals of Statistics*, 39(2):887–930, 2011.

[55] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[56] Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. *arXiv preprint arXiv:1611.01232*, 2016.

[57] Hakim Sidahmed, Samrat Phatale, Alex Hutcheson, Zhuonan Lin, Zhang Chen, Zac Yu, Jarvis Jin, Simral Chaudhary, Roman Komarytsia, Christiane Ahlheim, et al. Parameter efficient reinforcement learning from human feedback. *arXiv preprint arXiv:2403.10704*, 2024.

[58] Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. Improving lora in privacy-preserving federated learning. *arXiv preprint arXiv:2403.12313*, 2024.

[59] Tian Tong, Cong Ma, and Yuejie Chi. Accelerating ill-conditioned low-rank matrix estimation via scaled gradient descent. *Journal of Machine Learning Research*, 22(150):1–63, 2021.

[60] Tian Tong, Cong Ma, and Yuejie Chi. Low-rank matrix recovery with scaled subgradient methods: Fast and robust convergence without the condition number. *IEEE Transactions on Signal Processing*, 69:2396–2409, 2021.

[61] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[62] Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. *arXiv preprint arXiv:2210.07558*, 2022.

[63] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[64] Hongyi Wang, Saurabh Agarwal, Yoshiki Tanaka, Eric Xing, Dimitris Papailiopoulos, et al. Cuttlefish: Low-rank model training without all the tuning. *Proceedings of Machine Learning and Systems*, 5:578–605, 2023.

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

[66] Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. Lora-pro: Are low-rank adapters properly optimized? *arXiv preprint arXiv:2407.18242*, 2024.

[67] Zihan Wang, Deli Chen, Damai Dai, Runxin Xu, Zhuoshu Li, and Yu Wu. Let the expert stick to his last: Expert-specialized fine-tuning for sparse architectural large language models. *arXiv preprint arXiv:2407.01906*, 2024.

[68] Ziyao Wang, Zheyu Shen, Yexiao He, Guoheng Sun, Hongyi Wang, Lingjuan Lyu, and Ang Li. Flora: Federated fine-tuning large language models with heterogeneous low-rank adaptations. *arXiv preprint arXiv:2409.05976*, 2024.

[69] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

 https://doi.org/10.52202/085713-1258

[70] Xun Wu, Shaohan Huang, and Furu Wei. Mixture of lora experts. *arXiv preprint arXiv:2404.13628*, 2024.

[71] Wenhan Xia, Chengwei Qin, and Elad Hazan. Chain of lora: Efficient fine-tuning of language models via residual learning. *arXiv preprint arXiv:2401.04151*, 2024.

[72] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*, 2024.

[73] Xingyu Xu, Yandi Shen, Yuejie Chi, and Cong Ma. The power of preconditioning in overparameterized low-rank matrix sensing. In *International Conference on Machine Learning*, pages 38611–38654. PMLR, 2023.

[74] Greg Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019.

[75] Greg Yang and Edward J Hu. Feature learning in infinite-width neural networks. *arXiv preprint arXiv:2011.14522*, 2020.

[76] Greg Yang and Edward J Hu. Tensor programs iv: Feature learning in infinite-width neural networks. In *International Conference on Machine Learning*, pages 11727–11737. PMLR, 2021.

[77] Yang Yang, Wen Wang, Liang Peng, Chaotian Song, Yao Chen, Hengjia Li, Xiaolong Yang, Qinglin Lu, Deng Cai, Boxi Wu, et al. Lora-composer: Leveraging low-rank adaptation for multi-concept customization in training-free diffusion models. *arXiv preprint arXiv:2403.11627*, 2024.

[78] Can Yaras, Peng Wang, Laura Balzano, and Qing Qu. Compressible dynamics in deep overparameterized low-rank learning & adaptation. *arXiv preprint arXiv:2406.04112*, 2024.

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

[80] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

[81] Fangzhao Zhang and Mert Pilanci. Riemannian preconditioned lora for fine-tuning foundation models. *arXiv preprint arXiv:2402.02347*, 2024.

[82] Feiyu Zhang, Liangzhi Li, Junhao Chen, Zhouqiang Jiang, Bowen Wang, and Yiming Qian. Increlora: Incremental parameter allocation method for parameter-efficient fine-tuning. *arXiv preprint arXiv:2308.12043*, 2023.

[83] Jialun Zhang, Salar Fattahi, and Richard Y Zhang. Preconditioned gradient descent for over-parameterized nonconvex matrix factorization. *Advances in Neural Information Processing Systems*, 34:5985–5996, 2021.

[84] Jialun Zhang, Richard Y Zhang, and Hong-Ming Chiu. Fast and accurate estimation of low-rank matrices from noisy measurements via preconditioned non-convex gradient descent. In *International Conference on Artificial Intelligence and Statistics*, pages 3772–3780. PMLR, 2024.

[85] Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning. *arXiv preprint arXiv:2308.03303*, 2023.

[86] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023.

[87] Yuanhe Zhang, Fanghui Liu, and Yudong Chen. One-step full gradient suffices for low-rank fine-tuning, provably and efficiently. *arXiv preprint arXiv:2502.01235*, 2025.

[88] Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*, 2024.

[89] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023.

[90] Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. Opencodeinterpreter: Integrating code generation with execution and refinement. *arXiv preprint arXiv:2402.14658*, 2024.

[91] Bojia Zi, Xianbiao Qi, Lingzhi Wang, Jianan Wang, Kam-Fai Wong, and Lei Zhang. Delta-lora: Fine-tuning high-rank parameters with the delta of low-rank matrices. *arXiv preprint arXiv:2309.02411*, 2023.

 https://doi.org/10.52202/085713-1258

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have summarized our contribution at the end of the introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In Section 3.1 and 3.2, we discuss how to reduce the assumption of full-rank with weight decay, which makes our theory applicable in practice. As our algorithm involves the matrix inverse, we discuss the computational cost in the experimental study (see Section 5.1).

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Due to the page limitation of the main paper, we put the proof Section 3 and 4 into Appendix B and D, respectively. For clarity, we summarize the key steps for establishing the proof at the begining of Appendix B and D.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: In Section 5, we discuss the experimental setup, like hyperparameter choices and datasets, and provide the link to our code repository.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

https://doi.org/10.52202/085713-1258

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We have already provided the link to our code repository in Section 5.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We present the experimental setup for training and evaluation in the main text. Additional details about the datasets used for each task are provided in Appendix E.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer:[Yes]

   Justification: We report the evaluation score with mean and standard error for each experiment. The randomness of our experiment is discussed in Appendix E.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Experiment Setup in Section 5.1 and 5.2.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer:[Yes]

   Justification: We keep the code to preserve anonymity.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Our theoretical analysis would provide new insight for the broader context of low-rank matrix estimation (see Section 3.1). And for the practical impact, our work would be applied to another parameter-efficient setting easily, as we discuss in Appendix C.1.

    Guidelines

    - The answer NA means that there is no societal impact of the work performed.

https://doi.org/10.52202/085713-1258

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have gotten the permission to use the pretrained models, like Llama-3-8B and Llama-3.1-8B via Hugging face and have cited their work properly.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release new code implementing the proposed method. The code is documented and includes instructions for reproducibility. An anonymized version is provided for review.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve any crowdsourcing experiments or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve research with human subjects or crowdsourcing, and therefore no IRB approval was required.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

https://doi.org/10.52202/085713-1258

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Related Work

Low-rank adaptation(LoRA)([25]) has been the subject of extensive research in foundation models([51, 5, 1, 33, 55, 61]), with numerous variations and improvements ([34, 28, 32, 78, 12, 27, 91]). One line of research focuses on dynamically adjusting the LoRA rank during training. This includes DyLoRA[62], IncreLoRA[82], and AdaLoRA[86]. Another line of work involves enhancing LoRA performance through the addition of extra scaling matrices, which include DoRA[41] and DeepLoRA[78]. These directions are orthogonal to our work. Regarding the optimization of LoRA, we find that the following topics are close to our work.

Stable Feature Learning Under the infinite-width NN setting([19, 56]), LoRA+([21]) finds that the standard LoRA is inefficient and they propose to use different learning rates for A and B. To provide a careful choice of hyperparameters for efficient use of LoRA, a line of work analyzes LoRA under efficient learning ([20, 81]). Noticeably, [81] introduces preconditioners under a Riemannian metric ([45]) and updates LoRA by using scaled gradients of A and B simultaneously. While their method aims to improve stability and efficiency, it is important to note that their goal is not to approximate the full gradient. This approach does not yield an optimal approximation to the full gradient update. Moreover, [79] proposes an adaptive matrix preconditioning method preserving transformation-invariant, a sufficient condition for stable feature learning.

Approximation full-tuning or full gradient To fill the gap between LoRA and full fine-tuning, there are two lines of work with different motivations. The first class of work focuses on the initialization, like [66]. It proposes to make the initialization of LoRA align with the full-finetuning directly. However, after the first step, how difference between LoRA and full-tuning is unknown. The second line of work focuses on optimizing LoRA properly over the optimization trajectory([66, 50, 87]). Noticeably, [66] proposes to optimize the gradients of A and B together to approximate the full gradient. But the optimal approximation is hard to find under practical conditions and aligning momentum towards the full gradient requires storing a full-size matrix ($k \times d$) in their algorithm. These challenges also exist in later work ([50]).

Gradient Projection in LoRA Motivated by the view that LoRA updates can be viewed as performing random projection from the full gradient, F-LoRA([18]) achieves high-rank updates by resampling the projection matrices. There are also some approaches that propose training networks with low-rank factorized weights from scratch ([64, 32]). Random projection is also applied in Ga-LoRA([88]) and following work([39, 9]), but they need to access the full model and can't store the low-rank adapter in the end. On the contrary, without full-parameter learning, we use gradient projection to keep the gradient best preserved in the low-rank spaces.

Alternating Update To the best of our knowledge, we haven't found the existing work of updating LoRA alternately in the centralized setting, but in the decentralized setting, i.e., Federated Learning, we notice [7] used the alternating strategy to address the challenge of inaccurate model aggregation([68, 3, 58]) with computational and communication efficiency. Besides, in the centralized setting, [85] proposes to freeze A and update B, which would be regarded as a specific case of our work to do alternating minimization.

Scaled Gradient Descent Our proposed methods are also closely related to scaled gradient descent(Scaled GD) in traditional low-rank matrix estimation under over-parameterization and ill-conditioning ([59, 60, 29, 46]). Notably, [59] shows that the scaled GD would keep the convergence independent of the condition number. Different variants of scaled GD have been proposed and studied in work ([73, 83, 10, 84]). For the alternating scaled GD, [30] finds that it would enable faster convergence with larger step sizes compared with scaled GD. And [42] provably shows that alternating scaled GD would achieve a linear convergence rate, starting from arbitrary random initialization.

B The Proof and Details in Section 3

In this section, we provide the formal proofs and detailed discussions supporting the results presented in Section 3. Specifically, Appendix B.1 presents the proof of Theorem 1, removes the full-rank assumption in Corollary 1 via weight decay. Appendix B.2 contains the proof of Theorem 2 and demonstrates how the full-rank assumption can similarly be relaxed using weight decay in Corollary 2.

 https://doi.org/10.52202/085713-1258

B.1 The Proof in Section 3.1

B.1.1 The Proof of Theorem 1

Proof. The first-order condition of Problem (4) yields

$$sB_t^T(sB_t\tilde{\nabla}_{A_t}L - \nabla_{W_t}L) = 0, \tag{14}$$

where s is a positive scaling factor. Then we can reorganize it and obtain

$$sB_t^T B_t\tilde{\nabla}_{A_t}L = B_t^T\nabla_{W_t}L. \tag{15}$$

As we assume the matrix B is full rank, it yields

$$\tilde{\nabla}_{A_t}L = \frac{1}{s}(B_t^T B_t)^{-1}B_t^T(\nabla_{W_t}L). \tag{16}$$

Furthermore, recalling the definition of the gradient of standard LoRA in (1), we obtain

$$\tilde{\nabla}_{A_t}L = \frac{1}{s}(B_t^T B_t)^{-1}B_t^T(\nabla_{W_t}L) = \frac{1}{s^2}(B_t^T B_t)^{-1}\nabla_{A_t}L. \tag{17}$$

Similarly, we can obtain the closed-form solution of $\tilde{\nabla}_{B_t}L$ in (8). □

B.1.2 Corollary 1 and Its Proof

Corollary 1. *For $B \in \mathbb{R}^{k\times r}$ and $A \in \mathbb{R}^{r\times d}$, solving problems in (18)*

$$\begin{aligned}
&\min_{\tilde{\nabla}_{A_t}L} \|sB_t(\tilde{\nabla}_{A_t}L) - \nabla_{W_t}L\|_F^2 + \frac{\lambda}{2}\|s\tilde{\nabla}_{A_t}L\|_F^2 \\
&\min_{\tilde{\nabla}_{B_t}L} \|s(\tilde{\nabla}_{B_t}L)A_{t+1} - \nabla_{W_{t+\frac{1}{2}}}L\|_F^2 + \frac{\lambda}{2}\|s\tilde{\nabla}_{B_t}L\|_F^2,
\end{aligned} \tag{18}$$

yields the unique closed-form solution

$$\begin{aligned}
\tilde{\nabla}_{A_t}L &= \frac{1}{s}(B_t^T B_t + \lambda\mathbb{I}_{r\times r})^{-1}B_t^T(\nabla_{W_t}L) = \frac{1}{s^2}(B_t^T B_t + \lambda\mathbb{I}_{r\times r})^{-1}\nabla_{A_t}L, \\
\tilde{\nabla}_{B_t}L &= \frac{1}{s}(\nabla_{W_{t+\frac{1}{2}}}L)A_{t+1}^T(A_{t+1}A_{t+1}^T + \lambda\mathbb{I}_{r\times r})^{-1} = \frac{1}{s^2}\nabla_{B_t}L(A_{t+1}A_{t+1}^T + \lambda\mathbb{I}_{r\times r})^{-1}.
\end{aligned} \tag{19}$$

where $\mathbb{I}_{r\times r}$ is the $r \times r$ identity matrix and $\lambda > 0$.

Proof. For the first line problem in (18), the first-order condition yields

$$sB_t^T(sB_t\tilde{\nabla}_{A_t}L - \nabla_{W_t}L) + \lambda s^2\tilde{\nabla}_{A_t}L = 0, \tag{20}$$

where s is a positive scaling factor. Then we can reorganize it and obtain

$$s(B_t^T B_t + \lambda\mathbb{I})\tilde{\nabla}_{A_t}L = B_t^T\nabla_{W_t}L. \tag{21}$$

To keep $(B_t^T B_t + \lambda\mathbb{I})$ invertible, we only require that λ isn't too small and it yields

$$\tilde{\nabla}_{A_t}L = \frac{1}{s}(B_t^T B_t + \lambda\mathbb{I})^{-1}B_t^T(\nabla_{W_t}L). \tag{22}$$

Furthermore, recalling the definition of the gradient of standard LoRA in (1), we obtain

$$\tilde{\nabla}_{A_t}L = \frac{1}{s}(B_t^T B_t + \lambda\mathbb{I})^{-1}B_t^T(\nabla_{W_t}L) = \frac{1}{s^2}(B_t^T B_t + \lambda\mathbb{I})^{-1}\nabla_{A_t}L. \tag{23}$$

Similarly, we can obtain the closed-form solution of $\tilde{\nabla}_{B_t}L$ in (19). Noticeably, the result $(\nabla_{W_{t+\frac{1}{2}}}L)A_{t+1}^T = \nabla_{B_t}L$ holds with the fact that $W_{t+\frac{1}{2}} = W_0 + B_t A_{t+1}$. □

In Corollary 1, the hyperparameter λ can be small enough ($1e^{-6}$ in our numerical studies) and we don't tune the hyperparameter overall. For more discussion about the selection of λ in the over-parameterized setting for low-rank matrix estimation, please refer to APGD([42]), ScaledGD([73]), and NoisyPrecGD([84]).

B.2 Proof of Section 3.2

B.2.1 Proof of Theorem 2

Proof. The proof is similar to Theorem 1 thus we omit it here. □

B.2.2 Corollary 2 and Its Proof

Corollary 2. *If we assume M_t^B has aligned with the full gradient in the t-th iteration, by minimizing the following problem*

$$\min_{\tilde{M}_t^B} \|M_t^B A_t - \tilde{M}_t^B A_{t+1}\|_F^2 + \frac{\lambda}{2}\|\tilde{M}_t^A\|_F^2, \tag{24}$$

we can find the unique solution $\tilde{M}_t^B = M_t^B A_t A_{t+1}^T (A_{t+1} A_{t+1}^T + \lambda \mathbb{I})^{-1}$, which is the best approximation of current full gradient.

Proof. The proof is similar to Corollary 1 thus we omit it here. □

C Appendix for Algorithm 1

C.1 The Implementing Details for Algorithm 1

AltLoRA, as a novel PEFT method, can be seamlessly integrated into popular libraries such as Hugging Face Transformers [69]. The key engineering modifications are as follows:

- **Alternating Updates**: To enable alternating optimization of LoRA parameters, we extend the existing Transformer architecture by introducing a control argument within the `training_step` function. This argument identifies the current update phase and selectively disables gradient computation for parameters named `"lora_A"` or `"lora_B"`, thereby facilitating an efficient alternating update mechanism.

- **Custom Optimizer Integration**: Similar to prior LoRA variants that incorporate new optimizers [81, 66], AltLoRA can be easily adapted by implementing a new optimizer class. This allows flexible modification of the optimization dynamics tailored to the alternating update strategy. It would provide a broader impact to incorporate with other parameter-efficient structures, like MoE or RLHF, when using low-rank adaptation.

C.2 AltLoRA+

With the goal of approximating the full gradient under the memory constraint of standard LoRA, we propose AltLoRA in Algorithm 1 to properly optimize the training paradigm of LoRA. Furthermore, the ultimate goal is to fill the gap of performance between the existing parameter-efficient fine-tuning methods, like LoRA([25]), and the full model fine-tuning. Therefore, witnessing the success of incorporating the second momentum for accelerating and stabilizing the optimizing paradigm [25], we propose a variant of AltLoRA, called AltLoRA+ (see Algorithm 2) to help accelerate our optimizer with second momentum. The increasing memory cost for storing second momentum is $\mathcal{O}(kr + dr)$, so AltLoRA+ won't require storing the full size matrix $\mathcal{O}(kd)$ like LoRA-Pro [66].

 https://doi.org/10.52202/085713-1258

Algorithm 2: AltLoRA+: AltLoRA with Second Order Momentum

Input: Momentum states M_0^A, M_0^B, V_0^A and V_0^B, scaling factor $s = \frac{\alpha}{r}$, learning rate η, momentum coefficient β_1 and β_2, total number of steps T, weight decay coefficient γ, and constant ϵ

Output: Final matrices A_T and B_T

for $t = 0, \ldots, T-1$ **do**

  if $t \bmod 2 = 0$ **then**

    Update A**:**

      Only backpropagate w.r.t. A_t and obtain $\nabla_{A_t} L$

      $\tilde{\nabla}_{A_t} L = \frac{1}{s^2}(B_t^\top B_t)^{-1} \nabla_{A_t} L$

      $\tilde{M}_t^A = (B_t^\top B_t)^{-1} B_t^\top B_{t-1} M_{t-1}^A$

      $M_t^A \leftarrow \beta_1 \tilde{M}_t^A + (1-\beta_1)\tilde{\nabla}_{A_t} L$

      $V_t^A \leftarrow \beta_2 V_{t-1}^A + (1-\beta_2)(\tilde{\nabla}_{A_t} L \odot \tilde{\nabla}_{A_t} L)$

      $A_{t+1} \leftarrow A_t - \eta \left(M_t^A \oslash (\sqrt{V_t^A} + \epsilon) + \gamma A_t\right)$

  else

    Update B**:**

      Only backpropagate w.r.t. B_t and obtain $\nabla_{B_t} L$

      $\tilde{\nabla}_{B_t} L = \frac{1}{s^2} \nabla_{B_t} L (A_{t+1} A_{t+1}^\top)^{-1}$

      $\tilde{M}_t^B = M_{t-1}^B A_t A_{t+1}^\top (A_{t+1} A_{t+1}^\top)^{-1}$

      $M_t^B \leftarrow \beta_1 \tilde{M}_t^B + (1-\beta_1)\tilde{\nabla}_{B_t} L$

      $V_t^B \leftarrow \beta_2 V_{t-1}^B + (1-\beta_2)(\tilde{\nabla}_{B_t} L \odot \tilde{\nabla}_{B_t} L)$

      $B_{t+1} \leftarrow B_t - \eta \left(M_t^B \oslash (\sqrt{V_t^B} + \epsilon) + \gamma B_t\right)$

D Proof and Details of Section 4

In this section, we will start to analyze the training paradigm of AltLoRA in Algorithm 1 and AltLoRA+ in Algorithm 2. In Appendix D.1, we first give the formal definition of stable feature learning in Definition 2. Then we will analyze our methods without momentum on a toy model in Appendix D.1.1. Furthermore, in Appendix D.1.2, we provably show that AltLoRA or AltLoRA+ with arbitrary LoRA ranks achieves stable feature learning in the infinite dimension NN setting. Then, in Appendix D.2, we provably show that AltLoRA would achieve transformation invariance. Finally, in Appendix D.3, within an over-parameterized two-layer ReLU NN tuning problem, we prove that AltLoRA or AltLoRA+ without momentum would converge linearly without the requirement of spectral initialization.

D.1 Appendix for Section 4.1

First, let's recall the definition of stable feature learning below.

Definition 2 (Stable Feature Learning (Definition A.1.[81]))**.** *Consider any general LoRA layer BAx with $B \in \mathbb{R}^{k\times r}$ and $A \in \mathbb{R}^{r\times d}$ being LoRA parameters. Denote $\Delta_t = W_t - W_{t-1} = B_t A_t x - B_{t-1} A_{t-1} x$ for fine-tuning step t. We say that LoRA mdoel achieves Stable Feature Learning when x, Ax, $BAx \in \mathcal{O}(1)$ for alll LoRA layers and $\Delta_t \in \mathcal{O}(1)$ for all fine-tuning step t.*

D.1.1 Analysis on A Toy Model

Following LoRA+([21]), let's consider the simple linear model first

$$f(x) = (W + ba^T)x, \tag{25}$$

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 $a \sim \mathcal{N}_n(0, \sigma^2 \mathbf{I}_n), b \sim \mathcal{N}(0, \sigma_b^2)$. Conventionally, ba^T is initialized at zero for LoRA, and we thus consider setting $\sigma_a^2 = 0$, $\sigma_b^2 = \mathcal{O}(1)$.

For simplicity, assume AltLoRA or AltLoRA+ without momentum updates with learning rate $\eta = \mathcal{O}(n^c)$ for some $c \in \mathbb{R}$. Since the training process involves only elementary algebraic operations, the quantities there should be of powers of n. If we treat updates A and B each time as a single iteration, in iteration t, the feature update is given by

$$\begin{aligned}\Delta f_{t+1} :&= f_{t+1}(x) - f_t(x) \\ &= \left(b_t a_t^T - \eta b_t (\tilde{\nabla}_{a_t} L)^T - \eta (\tilde{\nabla}_{b_t} L) a_{t+1}^T\right) x - b_t a_t^T x \\ &= -\eta (f_t(x) - y)\|x\|^2 - \eta (a_{t+1}^T x)^2 (f_{t+\frac{1}{2}}(x) - y)\|a_{t+1}\|^{-2},\end{aligned} \tag{26}$$

where $f_{t+\frac{1}{2}}(x) := (W + b_t a_{t+1}^T)x$. We denote $\delta_t^1 = \eta b_t^2 (f_t(x) - y)\|x\|^2$, $\delta_t^2 = \eta (a_{t+1}^T x)^2 (f_{t+\frac{1}{2}}(x) - y)$. To achieve stable feature learning, it requires $\delta_t^1, \delta_t^2 \in \mathcal{O}(1)$ and further $f_t(x) \in \mathcal{O}(1)\ \forall t > 0$. Thus, we have the below modified linear constraints.

$$\begin{cases} c + 1 = 0 & (\text{for } \delta_t^1 = \Theta(1)), \\ c + 2\gamma[a_{t+1}^T x] - \gamma[\|a_{t+1}\|^2] = 0 & (\text{for } \delta_t^2 = \Theta(1)), \\ \gamma[b_{t+1}] + \gamma[a_{t+1}^T x] = 0 & (\text{for } f_{t+1}(x) = \Theta(1)), \end{cases} \tag{27}$$

where, for the sake of notational clarity, we introduce new notation γ such that $v = \mathcal{O}(n^{\gamma[v]})$ captures the polynomial behavior for any v.

Solving the equations in (27), we can derive $c = -1$. With $\eta = \mathcal{O}(n^{-1})$, we get $\gamma[b_1] = \gamma[b_0] = 0$ and $\gamma[a_1^T x] = \gamma[\eta b_0^{-1} y \|x\|^2]$. Recursively, we can derive $b_t, a_t, \delta_t^1, \delta_t^2 \in \mathcal{O}(1)$ for all t. Therefore, we obtain $f_t \in \mathcal{O}(1)$ and $\Delta f_t \in \mathcal{O}(1)$. The above toy model illustrates that our proposed method achieve stable learning with learning rates for A and B of the same order of magnitude.

D.1.2 Proof for Theorem 3

In this part, we extend the analysis above to a general neural architecture with LoRA layers. We show that the conclusion from the analysis on the linear model hold for general neural architecture.

Assumption 1 (Assumption 1 in [21])**.** *We assume that the gradient processing step by AltLora in Algorithm 1 (or AltLoRA+ in Algorithm 2) satisfies $g_A^t = \mathcal{O}(n)$ for all t where g_A^t is the processed gradient of A by AltLoRA (or AltLoRA+) in t-th update.*

Lemma 2 (Lemma A.3. in [81])**.** *For any matrix $A \in \mathbb{R}^{m \times n}$, where m being powers of n, such that $A^\top A$ is invertible and $\gamma[A_{ij}] = c$ for all (i, j), we have*

$$\gamma\left[(A^\top A)^{-1}\right] = -\gamma[\|a\|^2]$$

with a being any column of A.

Now, we state the formal version of our Theorem 2.

Theorem 5. *Let g_t^A and g_t^B denote the processed gradient of A and B, respectively, in Algorithm 1 or Algorithm 2. Assume Assumption 1 holds for the gradient processing of AltLoRA or AltLoRA+. And g_t^A and $g_t^B \in \mathcal{O}(1)$ after the gradient processed. Further assume BAx has dimension of $\mathcal{O}(n)$. Then the following results hold:*

(1) AltLoRA (AltLoRA+) achieves stable feature learning with $\eta = \mathcal{O}(1)$.

(2) If we consider AltLoRA or AltLoRA+ without momentum, the update yields

$$W_{t+1} = W_t - \eta Proj_{c(B_t)}(\nabla_{W_t} L) - \eta (\nabla_{W_{t+\frac{1}{2}}} L) Proj_{r(A_{t+1})}, \tag{28}$$

where $\eta Proj_{c(B_t)}(\nabla_{W_t} L), \eta (\nabla_{W_{t+\frac{1}{2}}} L) Proj_{r(A_{t+1})} \in \mathcal{O}(1)$. However, when doing joint update, the update will introduce additional across term $\eta^2 (\nabla_{W_t} L) A_t^T (A_t A_t^T)^{-1} (B_t^T B_t)^{-1} B_t^T (\nabla_{W_t} L) \in \mathcal{O}(1)$. The across term is indeed the second order term w.r.t η, but it is same magnitude as $\eta Proj_{c(B_t)}(\nabla_{W_t} L)$ and $\eta (\nabla_{W_t} L) Proj_{r(A)}$ in infinite-width NN setting.

https://doi.org/10.52202/085713-1258

Proof. (**Part 1**) First, we will prove AltLoRA (AltLoRA+) can achieve stable feature learning. The technical lemmas and assumptions used for proof are also well-adapted in [21, 81].

We will alternately update A first then update B. If we treat update A frist then update B as a single iteration, it could yield the update of the full model W as

$$\begin{aligned}\Delta^t &= B_tA_tx - B_{t-1}A_{t-1}x \\ &= B_tA_tx - B_{t-1}A_tx + B_{t-1}A_tx - B_{t-1}A_{t-1}x \\ &= (B_t - B_{t-1})A_tx + B_{t-1}(A_t - A_{t-1})x \\ &= -\gamma g_B^{t-1}(A_tA_t^\top)^{-1}A_tx - \gamma B_{t-1}(B_{t-1}^\top B_{t-1})^\gamma g_A^{t-1}x.\end{aligned} \tag{29}$$

Then we will denote these two parts of the update in the R.H.S of (29) as

$$\begin{aligned}\delta_1^t &= \eta B_{t-1}(B_{t-1}^\top B_{t-1})^\gamma g_A^{t-1}x \\ \delta_2^t &= \eta g_B^{t-1}(A_tA_t^\top)^{-1}A_tx.\end{aligned} \tag{30}$$

Following Assumption 1, we know $g_A^{t-1}x \in \mathcal{O}(n)$. Thus the conditions of δ_1^t, δ_2^t, $B_{t-1}A_tx \in \mathcal{O}(x)$ are equivalent to

$$\begin{aligned}&\gamma[\eta] + \gamma[B_{t-1}] + \gamma[(B_{t-1}^\top B_{t-1})^{-1}] + 1 = 0 \\ &\gamma[\eta] + \gamma[A_tA_t^\top] + \gamma[A_tx] = 0.\end{aligned} \tag{31}$$

For gradient update, we have

$$\begin{aligned}A_tx &= A_{t-1}x - \eta(B_{t-1}^\top B_{t-1})^{-1}g_A^{t-1}x \\ B_t &= B_{t-1} - \eta g_B^{t-1}(A_tA_t^\top)^{-1}.\end{aligned} \tag{32}$$

thus we have

$$\Rightarrow \begin{cases} \gamma[B_t] = \max\left\{\gamma[B_{t-1}], \gamma[\eta] + \gamma[(B_{t-1}^\top B_{t-1})^{-1}]\right\} \\ \gamma[A_tx] = \max\left\{\gamma[A_{t-1}x], \gamma[\eta] + \gamma[(B_{t[1}^\top B_{t-1})^{-1}] + 1\right\}. \end{cases}$$

Note $A_1 = A_0$, the recursive argument of δ_t^1 and $\delta_t^2 \in \mathcal{O}(1)$ is the same as [81]. Therefore, we find that AltLoRA or AltLoRA+ achieves stable feature learning with $\eta = \mathcal{O}(1)$. We can conclude that our algorithm would achieve stable feature learning with the same order of η in contrast to the standard LoRA ([21])

(**Part 2**) When removing the momentum in our methods, under Assumption 1, it would achieve stable feature learning as Part 1 has proved. Then the update of the full model W is

$$W_{t+1} = W_t - \eta Proj_{c(B_t)}(\nabla_{W_t}L) - \eta(\nabla_{W_{t+\frac{1}{2}}}L)Proj_{r(A_{t+1})}, \tag{33}$$

where $\eta Proj_{c(B_t)}(\nabla_{W_t}L), \eta(\nabla_{W_{t+\frac{1}{2}}}L)Proj_{r(A_{t+1})} \in \mathcal{O}(1)$.

However, when doing a joint update with scaled gradient descent ([81]), the update of the full model W is

$$\begin{aligned}W_{t+1} = W_t &- \eta Proj_{c(B_t)}(\nabla_{W_t}L) - \eta(\nabla_{W_t}L)Proj_{r(A_t)} \\ &+ \eta^2(\nabla_{W_t}L)A_t^T(A_tA_t^T)^{-1}(B_t^TB_t)^{-1}B_t^T(\nabla_{W_t}L)\end{aligned} \tag{34}$$

where the additional cross term $\eta^2(\nabla_{W_t}L)A_t^T(A_tA_t^T)^{-1}(B_t^TB_t)^{-1}B_t^T(\nabla_{W_t}L)$ is of order $\mathcal{O}(1)$. While this term is second-order with respect to η, it shares the same magnitude as the first-order terms $\eta Proj_{c(B_t)}(\nabla_{W_t}L)$ and $\eta(\nabla_{W_t}L)Proj_{r(A_t)}$ under the infinite-width neural network setting. A straightforward explanation is that the embedding dimension contributes quadratically to the cross term's effect, matching the overall scale of the first two terms. □

D.2 Proof of Section 4.2

First, let's restate Lemma 1 again and prove it.

Lemma 3. *If any two pairs of LoRA factors* $(A_1, B_1), (A_2, B_2)$ *satisfying*

$$W = W_0 + B_1A_1 = W_0 + B_2A_2, \tag{35}$$

then

$$\begin{aligned} Proj_{c(B_1)} &= Proj_{c(B_2)} \\ Proj_{r(A_1)} &= Proj_{r(A_2)} \end{aligned} \tag{36}$$

where $Proj_{c(\cdot)}$ *and* $Proj_{r(\cdot)}$ *is defined in (9).*

Proof. we know the column spaces of B_1 and B_2 are equivalent, as both of them span the column space of $W - W_0$. Thus, the projection matrices to the column spaces of B_1 and B_2 are the same, i.e., $Proj_{c(B_1)} = Proj_{c(B_2)}$, where $Proj_{c(\cdot)}$ is defined in (9). Similarly, the row spaces of A_1 and A_2 are equivalent. And the projection matrices to the column spaces of A_1 AND A_2 are the same, i.e., $Proj_{r(A_1)} = Proj_{r(A_2)}$. □

Lemma 1 tells that if two pairs of low-rank adaptation would get the same full model update, the projection matrix would preserve invariant over the pairs of low-rank adaptation. Next, we will restate Theorem 4 here and start to prove the theorem.

Theorem 6. *In Algorithm 1, every term is consistent across all equivalent LoRA pairs. Consequently, Algorithm 1 is transformation-invariant.*

Proof. Now we will use an inductive argument to prove it. Let's denote $(B_{1,t}, A_{1,t})$, $(B_{2,t}, A_{2,t})$ as two pairs of LoRA adaptation in the t-th interaction statisfying

$$W_0 + B_{1,t}A_{1,t} = W_0 + B_{2,t}A_{2,t}. \tag{37}$$

For the first pair $(B_{1,t}, A_{1,t})$, we denote $\tilde{M}^A_{1,t}$ and $M^A_{1,t}$ as the momentum used for $A_{1,t}$ in Algorithm 1. Let's assume, for the $(t-1)$-th iteration, we have the equivalent decomposition

$$B_{1,t-1}A_{1,t-1} = B_{1,t-1}, A_{1,t-1}. \tag{38}$$

Besides, we assume it is transformation invariance to $(t-1)$ iteration, then

$$B_{1,t-2}M^A_{1,t-2} = B_{2,t-2}M^A_{2,t-2} \tag{39}$$

$$M^B_{1,t-2}A_{1,t-2} = M^B_{1,t-2}A_{1,t-2}, \tag{40}$$

which implies that the historical information is invariant over the pairs of (B_1, A_1) and (B_2, A_2).

Then for the t-th iteration, we need to prove

$$B_{1,t-1}M^A_{1,t-1} = B_{2,t-1}M^A_{2,t-1} \tag{41}$$

$$M^B_{1,t-1}A_{1,t-1} = M^B_{2,t-1}A_{2,t-1}, \tag{42}$$

holds as well, and the update is transformation-invariant $B_{1,t}A_{1,t} = B_{2,t}A_{2,t}$.

First, we will focus on the update of A and prove $B_{1,t-1}M^A_{1,t-1} = B_{2,t-1}M^A_{2,t-1}$. Recalling the definition of $M^A_{1,t}$ is the cumulative gradient to the time t in Algorithm 1 , it yields

$$\begin{aligned} &B_{1,t-1}M^A_{1,t-1} \\ &= B_{1,t-1}\left(\beta_1(B^\top_{1,t-1}B_{1,t-1})^{-1}B^T_{1,t-1}B_{1,t-2}M^A_{1,t-2} + (1-\beta_1)\frac{1}{s^2}(B^\top_{1,t-1}B_{1,t-1})^{-1}\nabla_{A_{1,t-1}}L\right) \\ &= \beta_1 Proj_{c(B_{1,t-1})}B_{1,t-2}M^A_{1,t-2} + (1-\beta_1)\frac{1}{s^2}B_{1,t-1}(B^\top_{1,t-1}B_{1,t-1})^{-1}\nabla_{A_{1,t-1}}L \\ &= \beta_1 Proj_{c(B_{1,t-1})}B_{1,t-2}M^A_{1,t-2} + (1-\beta_1)\frac{1}{s}Proj_{c(B_{1,t-1})}\nabla_{W_{1,t-1}}L, \end{aligned} \tag{43}$$

 https://doi.org/10.52202/085713-1258

where the last line uses the results in (1) and $W_{1,t-1} := W_0 + B_{1,t-1}A_{1,t-1}$. Next, under the assumption for induction in (41) and Lemma 1, it yields

$$\begin{aligned} B_{1,t-1}M^A_{1,t-1} &= \beta_1 Proj_{c(B_{1,t-1})} B_{1,t-2}M^A_{1,t-2} + (1-\beta_1)\frac{1}{s} Proj_{c(B_{1,t-1})} \nabla_{W_{1,t-1}} L \\ &= \beta_1 Proj_{c(B_{2,t-1})} B_{2,t-2}M^A_{2,t-2} + (1-\beta_1)\frac{1}{s} Proj_{c(B_{2,t-1})} \nabla_{W_{2,t-1}} L \\ &= B_{2,t-1}M^A_{2,t-1}. \end{aligned} \tag{44}$$

After updating A, we can find the update of the full model as

$$\begin{aligned} B_{1,t-1}A_{1,t} &= B_{1,t-1}(A_{1,t-1} - \eta M^A_{1,t-1}) \\ &= B_{1,t-1}A_{1,t-1} - \eta B_{1,t-1}M^A_{1,t-1} \\ &= B_{2,t-1}A_{2,t-1} - \eta B_{2,t-1}M^A_{2,t-1} \\ &= B_{2,t-1}A_{2,t}, \end{aligned} \tag{45}$$

where the second-to-last line uses the results (38) in $(t-1)$-th iteration and the results in (44). Again, reapplying Lemma 1, we can find that $Proj_{c(A_{1,t})} = Proj_{c(A_{2,t})}$.

Up to now, we have shown that the update of A is transformation-invariant and $B_{1,t-1}M^A_{1,t-1} = B_{1,t-1}M^A_{1,t-1}$. With a similar argument, we can prove $M^B_{1,t-1}A_{1,t-1} = M^B_{1,t-1}A_{1,t-1}$ and $B_{1,t}A_{1,t} = B_{2,t}A_{2,t}$. Therefore, with the inductive argument, we prove the update of Algorithm 1 is transformation-invariant. □

In contrast to the prior work [79], our analysis centers on Lemma 1 to establish the proof of Theorem 4. Leveraging the alternating update strategy in Algorithm 1, we analyze the contributions of A and B to the full model update separately, allowing us to rigorously demonstrate transformation invariance. In comparison, [79] adopts a joint update of A and B, which introduces a cross term $\delta B \delta A$ that is ignored in their analysis, resulting in an inexact form of transformation invariance. Our alternating approach provides a principled direction toward achieving exact transformation invariance.

Discussion With our newly designed momentum mechanism, the first-order momentum terms remain consistent across all equivalent LoRA pairs, thereby ensuring that AltLoRA is robust to transformation invariance. In contrast, AltLoRA+ does not preserve this invariance. Motivated by this observation, we further attempt to design a second-order momentum mechanism that aligns optimally within the low-rank space under memory constraints. Although the second-order momentum terms are individually consistent across equivalent LoRA pairs, their combination with the first-order momentum leads to inconsistencies, ultimately breaking transformation invariance. To address this issue, employing unscaled gradients and momentum, as demonstrated by LoRA-Rite [79], could be a viable solution. However, as this approach diverges from our primary focus, we leave it for future work.

D.3 Convergence Analysis

D.3.1 Set Up

Following the previous work ([81]), we provide a convergence analysis of the proposed algorithm within the over-parameterized two-layer ReLU NN tuning problem. For a data matrix $X \in \mathbb{R}^{n\times d}$ and and any arbitrary vector u, we consider a set of diagonal matrices $\{\mathrm{diag}([Xu \geq 0]) \mid u \in \mathbb{R}^d\}$, which take value 1 or 0 along the diagonals that indicate the set of possible arrangement activation patterns for the ReLU activation. Let the distinct elements of this set be denoted as $D_1, \ldots, D_P$ (see [81] for more details). The constant P corresponds to the total number of partitions of $\mathbb{R}^d$ by hyperplanes passing through the origin that are also perpendicular to the rows of X [49]. Intuitively, P can be regarded as the number of possible ReLU activation patterns associated with X. [49] explains that a two-layer ReLU problem shares the same optimal objective with the convex problem

$$\min_{W_i\ i\in[P]} \frac{1}{2} \left\| \sum_{i=1}^{P} D_i X W_i - Y \right\|_F^2. \tag{46}$$

As we focus on fine-tuning, given a pretrained model with model weights $\{W_i\}_{i=1}^P$, we can do low-rank adaptation and rewrite the problem (46) as

$$\min_{A_i,B_i,i=1,\cdots P} \frac{1}{2}\left\|\sum_{i=1}^P D_i X(W_i + B_i A_i) - Y\right\|_F^2, \tag{47}$$

where $X \in \mathbb{R}^{n\times d}$, $A_i \in \mathbb{R}^{r\times c}$, $B_i \in \mathbb{R}^{d\times r}$ and $Y \in \mathbb{R}^{n\times c}$. We consider the response model $Y = \sum_i^P D_i X(W_i + B_i^\star A_i^\star)$. We define $X^\star := \sum_i^P B_i^\star A_i^\star$ are fixed and unknown matrices. Let's denote $\sigma_r(\cdot)$ as the r-th largest singular value. First let's introduce the definition of Restricted Isometry Property (RIP).

Definition 3. *(Restricted Isometry Property, [53]) The matric $C \in \mathbb{R}^{n\times d}$ is said to satisfy Restricted Isometry Property(RIP) with parameters (r, δ_r) if there exists constants $0 \le \delta_r \le 1$, for any matrices $M \in \mathbb{R}^{d\times c}$ with rank r, the below holds*

$$(1-\delta_r)\|M\|_F^2 \le \|CM\|_F^2 \le (1+\delta_r)\|M\|_F^2. \tag{48}$$

RIP is a widely used condition in the filed of compressed sensing ([42, 15, 53, 73]), which states that the operator C approximately preserves distances between low-rank matrices. In the absence of noise, we can establish a direct relationship between the loss function and the recovery error. If we denote $C_i := D_i X$, Problem (47) is equivalent to the problem below up to a change of labels

$$\min_{A_i,B_i,i=1,\cdots P} L_c(\boldsymbol{B}, \boldsymbol{A}) := \frac{1}{2}\left\|\sum_{i}^P C_i(B_i A_i - X_\star)\right\|_F^2, \tag{49}$$

where $\boldsymbol{B} = \{B_1, \cdots, B_P\}$ and $\boldsymbol{A} = \{A_1, \cdots, A_P\}$.

Notation Inspired by the previous work [42, 83, 84], we introduce two local norms and their corresponding dual norms for a matrix $W \in \mathbb{R}^{k\times r}$

$$\begin{aligned} &P_{A_t^i} := A_t^i (A_t^i)^T, \quad \|W\|_{P_{A_t^i}} := \|W P_{A_t^i}^{\frac{1}{2}}\|_F, \quad \|W\|_{P^\star_{A_t^i}} := \|W P_{A_t^i}^{-\frac{1}{2}}\|_F, \\ &P_{B_t^i} := (B_t^i)^T B_t^i, \quad \|W\|_{P_{B_t^i}} := \|W P_{B_t^i}^{\frac{1}{2}}\|_F, \quad \|W\|_{P^\star_{A_t^i}} := \|W P_{B_t^i}^{-\frac{1}{2}}\|_F. \end{aligned} \tag{50}$$

Here, we assume A_t^i and B_t^i are of full rank r for any i. If they aren't of full rank, we can replace them with the Moore-Penrose inverse([4]). Now we are ready to establish the convergence analysis.

D.3.2 Useful Lemma

For the t-th iteration, let's denote $\boldsymbol{B}_t = \{B_t^1, \cdots, B_t^P\}$ and $\boldsymbol{A}_t = \{A_t^1, \cdots, A_t^P\}$. If we apply AltLoRA or AltLoRA+ without momentum for Problem (49), for any $i \in [P]$, the alternating update rule as we proposed can be written as

$$\begin{aligned} A_{t+1}^i &\leftarrow A_t^i - \eta (B_t^i (B_t^i)^T)^{-1} \nabla_{A_t^i} L_c(\boldsymbol{B}_t, \boldsymbol{A}_t) \\ B_{t+1}^i &\leftarrow B_t^i - \eta \nabla_{B_t^i} L_c(\boldsymbol{B}_t, \boldsymbol{A}_{t+1}) ((A_{t+1}^i)^T A_{t+1}^i)^{-1}. \end{aligned} \tag{51}$$

First, we will list some assumptions used in our analysis.

Assumption 2. *Suppose that $C_i = D_i X$ obeys the r-RIP with a constant δ_r for each i.*

Assumption 3. *Suppose that $\|C_i^T C_j\|_2 := \|X^T D_i^T D_j X\|_2 \le \frac{1+\delta_r}{P(P-1)}$*

Assumption 2 and 3 also adopt in [81] to analyze their optimizer for LoRA. 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_r when $\|D_i\|_0$ is on the order of $r(d+c)/(d\delta_r^2)$. Note $\|X^T D_i^T D_j X\|_2 \le \|X^T X\|_2$ for all $(i,j)'s$. Thus bounding $\|X^T D_i^T D_j X\|_2$ amounts to bounding the largest singular value of the empirical covariance.

Lemma 4. *For a given $i \in [P]$, the gradient of Problem (49) are*

$$\begin{aligned} \nabla_{A_t^i} L(\boldsymbol{B}, \boldsymbol{A}) &= \sum_j^P (B_t^i)^T (C_i)^T C_j (B_t^j A_t^j - X_\star) \\ \nabla_{B_t^i} L(\boldsymbol{B}, \boldsymbol{A}) &= \sum_j^P (C_i)^T C_j (B_t^j A_{t+1}^j - X_\star)(A_{t+1}^j)^T. \end{aligned} \tag{52}$$

https://doi.org/10.52202/085713-1258

Proof. For any given i and t, it yields

$$\nabla_{A_t^i} L(\boldsymbol{B}, \boldsymbol{A}) = \frac{\partial}{\partial A_t^i} \left\{ \frac{1}{2} \left\| \sum_j^P C_j (B_j A_j - X_\star) \right\|_F^2 \right\} = \sum_j^P (B_t^i)^T (C_i)^T C_j (B_t^j A_t^j - X_\star). \quad (53)$$

Similarly, we can derive the $\nabla_{B_t^i} L(\boldsymbol{B}, \boldsymbol{A})$ as shown in (52). □

Lemma 5. *Suppose Assumption 2 and 3 holds, then we have*

$$\begin{aligned} L_c(\boldsymbol{B}_t, \boldsymbol{A}_{t+1}) &\leq L_c(\boldsymbol{B}_t, \boldsymbol{A}_t) - c_1 \max_i \left\| \nabla_{A_t^i} L_c(\boldsymbol{B}_t, \boldsymbol{A}_t) \right\|_{P^\star_{B_t^i}}^2 \\ L_c(\boldsymbol{B}_{t+1}, \boldsymbol{A}_{t+1}) &\leq L_c(\boldsymbol{B}_t, \boldsymbol{A}_{t+1}) - c_1 \max_i \left\| \nabla_{B_t^i} L_c(\boldsymbol{B}_t, \boldsymbol{A}_{t+1}) \right\|_{P^\star_{A_{t+1}^i}}^2, \end{aligned} \quad (54)$$

where $c_1 = P(\eta - \frac{\eta^2(1+\delta_r+\frac{1}{P})}{2})$.

Proof. Using the update rule in (51), we have

$$\begin{aligned} L_c(\boldsymbol{B}_t, \boldsymbol{A}_{t+1}) &= \frac{1}{2} \left\| \sum_i^P C_i (B_t^i A_{t+1}^i - X_\star) \right\|_F^2 \\ &= \frac{1}{2} \left\| \sum_i^P C_i \left(B_t^i \left(A_t^i - \eta ((B_t^i)^T B_t^i)^{-1} \nabla_{A_t^i} L_c(\boldsymbol{B}_t, \boldsymbol{A}_t) \right) - X_\star \right) \right\|_F^2 \\ &= \frac{1}{2} \left\| \sum_i^P C_i (B_t^i A_t^i - X_\star) \right\|_F^2 \\ &+ \underbrace{\frac{\eta^2}{2} \left\| \sum_i^P C_i B_t^i ((B_t^i)^T B_t^i)^{-1} \nabla_{A_t^i} L_c(\boldsymbol{B}_t, \boldsymbol{A}_t) \right\|_2^2}_{T_1} \\ &\quad \underbrace{- \eta \left\langle \sum_i^P C_i (B_t^i A_t^i - X_\star), \sum_i^P C_i B_t^i ((B_t^i)^T B_t^i)^{-1} \nabla_{A_t^i} L_c(\boldsymbol{B}_t, \boldsymbol{A}_t) \right\rangle}_{T_2} \end{aligned} \quad (55)$$

For T_1, recalling Lemma 4, then we have

$$\begin{aligned} T_1 &\leq \frac{\eta^2}{2} \sum_i^P \| C_i B_t^i ((B_t^i)^T B_t^i)^{-1} \nabla_{A_t^i} L_c(\boldsymbol{B}_t, \boldsymbol{A}_t) \|_F^2 \\ &+ \frac{\eta^2}{2} \sum_{i \neq j} \left\langle C_i B_t^i ((B_t^i)^T B_t^i)^{-1} \nabla_{A_t^i} L_c(\boldsymbol{B}_t, \boldsymbol{A}_t), C_j B_t^j ((B_t^j)^T B_t^j)^{-1} \nabla_{A_t^j} L_c(\boldsymbol{B}_t, \boldsymbol{A}_t) \right\rangle \\ &\overset{(a)}{\leq} \frac{\eta^2 (1+\delta_r)}{2} P \max_i \| \nabla_{A_t^i} L_c(\boldsymbol{B}_t, \boldsymbol{A}_t) \|_{P^\star_{B_t^i}}^2 \\ &+ \frac{\eta^2}{2} \max_{i \neq j} \| C_i^T C j \|_2 P(P-1) \max_i \| \nabla_{A_t^i} L_c(\boldsymbol{B}_t, \boldsymbol{A}_t) \|_{P_{B_t^i}}^2 \\ &\overset{(b)}{\leq} \frac{\eta^2 (1+\delta_r+\frac{1}{P})}{2} P \max_i \| \nabla_{A_t^i} L_c(\boldsymbol{B}_t, \boldsymbol{A}_t) \|_{P_{B_t^i}}^2, \end{aligned} \quad (56)$$

where (a) uses Cauchy Inequality, Assumption 2 and the fact that $\| B_t^i ((B_t^i)^T B_t^i)^{-\frac{1}{2}} \|_2^2 = 1$, (b) uses the assumption that $\max_{i \neq j} \| C_j^T C_j \|_2 \leq \frac{(1+\delta_r)}{P(P-1)}$.

For T_2, using Lemma 4 again, we have

$$\begin{aligned} T_2 &= \eta \left\langle \sum_j^P C_j(B_t^j A_t^j - X_\star), \sum_j^P C_j B_t^j ((B_t^j)^T B_t^j)^{-1} \nabla_{A_t^j} L_c(\boldsymbol{B}_t, \boldsymbol{A}_t) \right\rangle \\ &= \eta \sum_j^P \left\langle \sum_i^P C_i(B_t^i A_t^i - X_\star), C_j B_t^j ((B_t^j)^T B_t^j)^{-1} \nabla_{A_t^j} L_c(\boldsymbol{B}_t, \boldsymbol{A}_t) \right\rangle \\ &= \eta \sum_i^P \left\| \nabla_{A_t^i} L_c(\boldsymbol{B_t}, \boldsymbol{A}_t) \right\|_{P^\star_{B_t^i}}^2 \\ &\leq \eta P \max_i \|\nabla_{A_t^i} L_c(\boldsymbol{B}_t, \boldsymbol{A}_t)\|_{P_{B_t^i}}^2. \end{aligned} \tag{57}$$

To sum up, it yields

$$L_c(\boldsymbol{B}_t, \boldsymbol{A}_{t+1}) \leq L_c(\boldsymbol{B}_t, \boldsymbol{A}_t) - \left(\eta - \frac{\eta^2(1+\delta_r+\frac{1}{P})}{2}\right) P \max_i \left\| \nabla_{A_t^i} L_c(\boldsymbol{B_t}, \boldsymbol{A}_t) \right\|_{P^\star_{B_t^i}}^2. \tag{58}$$

Similarly, we can induce

$$L_c(\boldsymbol{B}_{t+1}, \boldsymbol{A}_{t+1}) \leq L_c(\boldsymbol{B}_t, \boldsymbol{A}_{t+1}) - \left(\eta - \frac{\eta^2(1+\delta_r+\frac{1}{P})}{2}\right) P \max_i \left\| \nabla_{B_t^i} L_c(\boldsymbol{B}_t, \boldsymbol{A}_{t+1}) \right\|_{P^\star_{A_{t+1}^i}}^2. \tag{59}$$

□

Lemma 6. *Suppose Assumption 2 holds, then, for any $i \in [P]$, we have*

$$\begin{aligned} \|\nabla_{A_t^i} L_c(\boldsymbol{B}_t, \boldsymbol{A}_t)\|_{P^\star_{B_t^i}}^2 &\geq 2(1-\delta_r) L_c(\boldsymbol{B}_t, \boldsymbol{A}_t) \\ \|\nabla_{B_t^i} L_c(\boldsymbol{B}_t, \boldsymbol{A}_{t+1})\|_{P^\star_{A_{t+1}^i}}^2 &\geq 2(1-\delta_r) L_c(\boldsymbol{B}_t, \boldsymbol{A}_{t+1}). \end{aligned} \tag{60}$$

Proof. See Lemma 6 in [42] for the detailed proof. □

Theorem 7. *Assume for any $i \in [p]$ the matrix $C_i = D_i X$ satisfies the rank r-RIP with constant δ_r (Assumption 2) and $0 \leq \eta \leq \frac{1}{1+\delta_r+\frac{1}{P}}$, then AltLoRA or AltLoRA+ without momentum solves the over-parameterized problem leads to*

$$L_c(\boldsymbol{B}_{t+1}, \boldsymbol{A}_{t+1}) \leq (1-\eta_c)^2 L_c(\boldsymbol{B}_t, \boldsymbol{A}_t) \tag{61}$$

and

$$\left\| \sum_i^P B_t^i A_t^i - X_\star \right\|_F^2 \leq \frac{1+\delta_r}{1-\delta_r} (1-\eta_c)^{2t} \left\| \sum_i^P B_0^i A_0^i - X_\star \right\|_F^2, \tag{62}$$

where $\eta_c = 2P(1-\delta_r)\left(\eta - \frac{\eta^2(1+\delta_r+\frac{1}{P})}{2}\right)$.

Proof.

$$\begin{aligned} L_c(\boldsymbol{B}_{t+1}, \boldsymbol{A}_{t+1}) &\leq L_c(\boldsymbol{B}_t, \boldsymbol{A}_{t+1}) - \left(\eta - \frac{\eta^2(1+\delta_r+\frac{1}{P})}{2}\right) P \max_i \left\| \nabla_{B_t^i} L_c(\boldsymbol{B}_t, \boldsymbol{A}_{t+1}) \right\|_{P^\star_{A_{t+1}^i}}^2 \\ &\leq L_c(\boldsymbol{B}_t, \boldsymbol{A}_{t+1}) - \left(\eta - \frac{\eta^2(1+\delta_r+\frac{1}{P})}{2}\right) 2P(1-\delta_r) L_c(\boldsymbol{B}_t, \boldsymbol{A}_{t+1}) \\ &\leq \left(1 - 2P(1-\delta_r)\left(\eta - \frac{\eta^2(1+\delta_r+\frac{1}{P})}{2}\right)\right) L_c(\boldsymbol{B}_t, \boldsymbol{A}_{t+1}) \\ &\leq (1-\eta_c)^2 L_c(\boldsymbol{B}_t, \boldsymbol{A}_t), \end{aligned} \tag{63}$$

 https://doi.org/10.52202/085713-1258

where we apply Lemma 5 and 6 and $\eta_c = 2P(1-\delta_r)\left(\eta - \frac{\eta^2(1+\delta_r+\frac{1}{P})}{2}\right)$. Moreover, under Assumption 2, we have

$$\left\|\sum_i^P B_t^i A_t^i - X_\star\right\|_F^2 \leq \frac{1+\delta_r}{1-\delta_r}(1-\eta_c)^{2t}\left\|\sum_i^P B_0^i A_0^i - X_\star\right\|_F^2. \tag{64}$$

□

E Appendix for Expirments

E.1 Details and Results for Supervised Fine-tuning

For the experimental setup, we follow the configuration used in LoRA-Pro [66] and summarize the key description here. As the experiments involve randomness from initialization and optimization, all results are averaged over three different random seeds.

Dialogue Generation Task We fine-tune large language models on a 52k subset of the WizardLM dataset [72] and evaluate it using the MT-Bench dataset [89]. GPT-4o is used to asses the quality of the model's response and we report the first-turn score as the metric.

Math Task We fine-tuning large language models on a 100k sample from the MetaMathQA dataset [80]. The model is then evaluated on the GSM8K test set [11], and we report the accuracy as the metric.

Coding Task We fine-tuning large language models on a 100k subset of the CodeFeedBack dataset [90] and test it on the HumanEval dataset [6], reporting the PASS@1 metric.

For the choice of learning rate, we perform grid search for LoRA, its variants, and AltLoRA+ over 1e-5, 4e-5, 1e-4. Since AltLoRA does not use second-moment estimates, we conduct an extended grid search over 1e-2, 1e-3, 1e-4, 4e-5, 1e-5. We observe that AltLoRA performs better with higher learning rates, and therefore report results using 1e-2, 1e-3, 1e-4 in the main evaluation. We set the iteration number to be 1 and the max step is 3000 for each experiment.

E.2 Additional Results

In Table 5, we compare our method with existing approaches across the three tasks described on Llama-3-8B model. Our method further bridges the performance gap between LoRA and full fine-tuning.

Table 5: Comparison of different LoRA variants on MT-Bench, GSM8K, and HumanEval benchmarks (accuracy in %) on Llama-3-8B-Base.

Method	MT-Bench	GSM8K	HumanEval
PreTrain	5.63	49.96± 0.38	34.76±0.37
LoRA	6.20	62.11±0.13	37.71±0.12
AltLoRA	6.05	64.39±0.23	40.81±0.47
AltLoRA+	6.34	67.38±0.13	43.81±0.31

In Table 6, we report the training loss of AltLoRA and its baselines on GSM8K using Llama-3-8B-Base model. Our method not only converges to a lower training loss than full fine-tuning, but also does so more rapidly than most baselines.

Base Llama-3-8B base and Llama-3.1-8B base model, we also consider the instruct tuning model-Llama3-8B-Instruct model. We fine-tune it on the MetaMathQA dataset and evaluate it on the GSM8K test set, reporting accuracy as the primary metric. In Table 7, our results demonstrate that even when using a stronger pretrained model, low-rank adaptation can still yield performance gains. More importantly, both AltLoRA and AltLoRA+ consistently outperform the standard LoRA baseline. We will add the full comparison in the revised revision.

Table 6: Training loss on GSM8K (Llama-3.1-8B) across training steps (lower is better).

Method	step 500	step 1000	step 1500	step 2000	step 2500	step 3000
Full Fine-tuning	0.51	0.28	0.24	0.18	0.18	0.18
LoRA	0.57	0.30	0.26	0.23	0.22	0.22
LoRA-Pro	0.51	0.31	0.24	0.22	0.21	0.21
LoRA-GA	0.45	0.24	0.20	0.18	0.18	0.18
LoRA-Rite	0.55	0.31	0.24	0.22	0.21	0.21
AltLoRA	0.48	0.25	0.21	0.19	0.18	0.18
AltLoRA+	0.45	0.23	0.19	0.18	0.17	0.17

Table 7: GSM8K accuracy (%) on Llama3-8B-instruct and its fine-tuned variants.

Method	GSM8K
Llama3-8B-instruct	61.24
+ *Full FT*	83.26
+ *LoRA*	81.12
+ *AltLoRA*	82.60
+ *AltLoRA+*	83.88

E.3 Additional Ablation Study

We conduct additional ablation studies to further demonstrate the practical effectiveness of our proposed methods. In Appendix E.3.1, we evaluate the performance of our methods under varying hyperparameter settings on the LLaMA 3-8B model. Furthermore, in Appendix E.3.2, beyond the learning rate, scaling factor α, and rank examined in Table 1, we perform comprehensive ablation studies for both AltLoRA and AltLoRA+ on the LLaMA 3.1-8B model.

E.3.1 Additional Ablation Study for Llama-3-8B Model

We further conduct ablation studies on the LLaMA 3-8B model to evaluate the robustness of our method under varying hyperparameter settings. As shown in Figure 2, we compare the performance of LoRA, AltLoRA, and AltLoRA+ on the GSM8K dataset across different learning rates and scaling factors $\alpha \in \{8, 16, 32\}$. AltLoRA+ consistently outperforms the baselines across all configurations, demonstrating both higher accuracy and stronger robustness to hyperparameter variation. We also have that all methods have better performance using $\alpha = 16$.

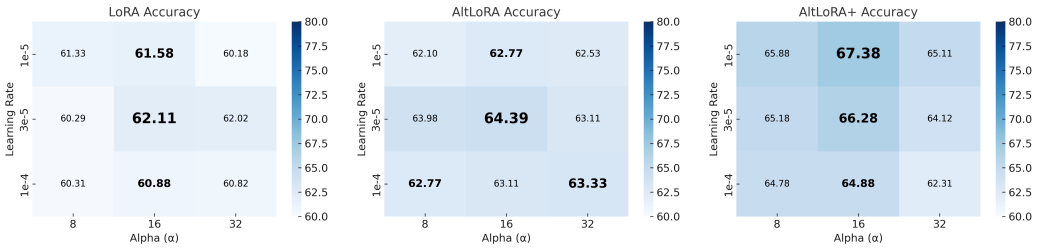

Figure 2: Evaluation Accuracy of LoRA, AltLoRA and AltLoRA+ for various learning rateη and scaling factor α combination on the GSM9K using Llama-3-8B.

E.3.2 Additional Ablation Study for Llama 3.1-8B Model

Training Epoch To rule out under-training as a potential factor for empirical performance, we increase the training budget to *5 epochs* under the same setup as Table 2. Results are reported in Table 8. (i) More epochs generally improve all methods; (ii) *AltLoRA* and *AltLoRA+* retain and even strengthen their advantage—*AltLoRA+* reaches $\sim$76% on GSM8K and approaches Full FT on

https://doi.org/10.52202/085713-1258

Table 8: Comparison of different LoRA variants on MT-Bench, GSM8K, and HumanEval benchmarks on Llama-3.1-8B-Base (5 training epoches). **Bold** indicates the best result, underline represents the second-best one.

Method	MT-Bench	GSM8K	HumanEval
Full FT	6.24±0.03	75.31±0.47	**50.42±1.20**
LoRA	6.19±0.02	66.78±1.14	41.54±1.01
PiSSA	6.11±0.03	67.39±0.61	42.12±1.45
rsLoRA	6.18±0.02	67.98±1.21	44.31±1.31
LoRA+	6.21±0.02	74.00±0.61	44.32±1.33
DoRA	6.23±0.03	67.12±0.84	45.12±1.32
AdaLoRA	6.05±0.02	68.29±0.98	42.58±1.84
LoRA-GA	6.24±0.05	73.44±1.10	45.56±1.48
LoRA-Pro	6.22±0.02	73.24±0.79	43.21±1.50
LoRA-Rite	6.20±0.03	73.48±0.65	45.46±1.21
AltLoRA	**6.26±0.04**	74.52±0.37	46.28±1.11
AltLoRA+	6.18±0.04	**76.01±0.34**	50.01±1.01

HumanEval; (iii) compared with standard LoRA and its variants (LoRA-Pro, LoRA-Rite, LoRA-GA), our methods benefit more consistently as the training budget grows, indicating that the gains are not a short-iteration artifact but persist under stronger training.

Ablation study on the updating strategy In Table 9, in contrast to joint update with scaled gradient descent [81], AltLoRA can optimally approximates the full gradient with alternating update and obtain better performance in evaluation. Interestingly, we find that the alternating update scheme—where matrix B is updated before A—consistently yields better performance. One possible explanation is that, under the standard initialization where B is set to zero, updating A first does not lead to meaningful descent.

Table 9: Performance comparison of LoRA, AltLoRA and AltLoRA+ on the GSM8K and Llama 3.1 8B with different updating strategies.

GSM8K	LoRA	AltLoRA	AltLoRA+
Alternating (A first)	66.11	74.49	76.91
Alternating (B first)	67.66	76.31	76.97
Joint Update	66.43	74.21	76.56

Ablation study on the number of LoRAs As low-rank adaptation comes to be a popular parameter-efficient technique for fine-tuning, it's well applied to more complicated scenarios ([43, 77, 57, 23, 70]). Notably, a very significant application is to improve the structure of the mixture of experts with parameter efficiency([70, 37]), handling multiple tasks simultaneously ([48, 26]) and addressing catastrophic forgetting ([14]). Following the work ([70]), we explore the performance as the number of LoRAs varies and utilize the gating balancing loss. Additionally, we compare AltLoRA and standard LoRA on the GSM8K dataset using the Llama 3.1-8B model(see Table 10). In our experiments, the number of LoRA experts is set to $\{1, 4, 8\}$, and the entropy regularization weight is 0.0001. We observe that increasing the number of LoRA experts enhances the capacity of the language model, leading to improved performance.

Table 10: Comparison of the mixture of experts model, with different expert numbers on GSM8K and Llama 3.1-8B-Base

Expert Num	LoRA	AdaLoRA	LoRA+	AltLoRA	AltLoRA+
1	66.11	72.63	72.33	74.49	76.91
4	67.43	71.71	71.27	75.01	77.33
8	67.89	70.34	71.44	75.33	76.94

Ablation Study on Initialization. To further validate the robustness of our method with respect to initialization, as discussed in Section 4.3, we conduct an ablation study using different initialization strategies. "Gaussian" refers to the standard random initialization used in the original LoRA framework [25]. "Kaiming" denotes the widely adopted Kaiming initialization, which is designed to maintain variance stability across layers. "Spectral" represents an initialization strategy based on spectral decomposition, where we perform singular value decomposition (SVD) on the pretrained weight matrix and construct the low-rank components using the top-r singular vectors, like the initialization proposed in [88]. In Table 11, we can see that with different initialization strategies, our method would achieve a superior performance over the standard LoRA. Without spectral initialization, using Kaiming initialization for A and setting B to be zero would achieve the best performance. Besides, to ensure the initial update of BA is zero, one of the matrices must be initialized to zero. Notably, setting $B = 0$ while using a small initialization for A yields better performance compared to the reverse setup. This finding is consistent with observations in existing literature [20].

Table 11: Comparison of the initialization strategies on GSM8K and Llama 3.1-8B-Base

Initialization Strategy		**LoRA**	**AltLoRA**	**AltLoRA+**
A	**B**			
Gaussian	zero	66.37	73.13	76.87
zero	Gaussian	66.18	72.13	76.50
Kaiming	zero	65.11	74.49	76.91
zero	Kaiming	67.10	74.03	76.88
Spectral	zero	67.63	74.67	76.60
zero	Spectral	67.10	74.61	76.37

E.4 The performance and efficiency trade-off

We summarize the memory–accuracy trade-off of our method against strong LoRA baselines on GSM8K with the Llama-3.1-8B backbone in Table 12. Under comparable memory budgets, **AltLoRA** consistently improves accuracy over standard LoRA and LoRA-Rite, and **AltLoRA+** achieves the best overall accuracy while maintaining a memory footprint close to standard LoRA. This demonstrates that our optimization yields a more favorable efficiency frontier: for a given memory cost, AltLoRA variants deliver higher accuracy; for a target accuracy, they require less memory.

Table 12: Memory cost (GB) vs. accuracy (%) on GSM8K (fine-tuning Llama-3.1-8B). AltLoRA and AltLoRA+ improve accuracy while preserving LoRA-level memory efficiency.

Method	**Memory Cost (GB)**	**Accuracy (%)**
Full FT	> 48	73.31
LoRA	22.26	66.11
LoRA-Rite	25.39	74.10
LoRA-Pro	40.12	73.12
AltLoRA	**22.56**	**74.49**
AltLoRA+	**23.16**	**76.91**

https://doi.org/10.52202/085713-1258