LoRA-Pro: Are Low-Rank Adapters Properly Optimized?

Zhengbo Wang[1,2], Jian Liang[2,3] *
[1] University of Science and Technology of China
[2] NLPR & MAIS, Institute of Automation, Chinese Academy of Sciences
[3] School of Artificial Intelligence, University of Chinese Academy of Sciences
zhengbowang@mail.ustc.edu.cn, liangjian92@gmail.com

Abstract

Low-Rank Adaptation, also known as LoRA, has emerged as a prominent method for parameter-efficient fine-tuning foundation models by re-parameterizing the original matrix into the product of two low-rank matrices. Despite its efficiency, LoRA often yields inferior performance compared to full fine-tuning. In this paper, we propose LoRA-Pro to bridge this performance gap.

Firstly, we delve into the optimization processes in LoRA and full fine-tuning. We reveal that while LoRA employs low-rank approximation, it neglects to approximate the optimization process of full fine-tuning. To address this, we introduce a novel concept called the "equivalent gradient." This virtual gradient makes the optimization process on the re-parameterized matrix equivalent to LoRA, which can be used to quantify the differences between LoRA and full fine-tuning. The equivalent gradient is derived from the gradients of matrices A and B. To narrow the performance gap, our approach minimizes the differences between the equivalent gradient and the gradient obtained from full fine-tuning during the optimization process. By solving this objective, we derive optimal closed-form solutions for updating matrices A and B. Our method constrains the optimization process, shrinking the performance gap between LoRA and full fine-tuning.

Extensive experiments on natural language processing tasks validate the effectiveness of our method.

1 Introduction

Foundational models [Radford et al., 2021, Brown et al., 2020, Achiam et al., 2023, Kirillov et al., 2023, Rombach et al., 2022] have become the cornerstone of modern deep learning. By undergoing pre-training on massive datasets, these models typically exhibit excellent generalization and versatility. Remarkably, some foundation models even demonstrate emergent properties [Hoffmann et al., 2022, Kaplan et al., 2020]. As a result, foundation models have been widely applied to various downstream applications.

Despite these advantages, the huge number of parameters in foundational models hinders their broader application. The substantial parameter count results in high fine-tuning costs for these tasks. To address this issue, recent research has focused on parameter-efficient fine-tuning (PEFT) methods [Hu et al., 2022, Houlsby et al., 2019, Lester et al., 2021, Zhou et al., 2022]. PEFT methods reduce the fine-tuning cost by keeping the foundation models frozen and only fine-tuning small, additional lightweight adapters. With the majority of parameters frozen, PEFT enables faster fine-tuning and requires fewer computational resources.

Low-rank adaptation [Hu et al., 2022], also known as LoRA, is one of the most famous PEFT methods, which has been widely adopted across various domains. Inspired by previous works [Aghajanyan et al., 2021, Li et al., 2018], LoRA hypothesizes that the changes in weights during model adaptation exhibit a low-rank structure. To capture this, LoRA re-parameterizes these changes by expressing them as the product of two low-rank matrices: $W = W_0 + \Delta W \approx W_0 + sBA$, where s is a scaling factor, and $A \in \mathbb{R}^{r\times n}$ and $B \in \mathbb{R}^{m\times r}$ are low-rank matrices with rank $r \ll \min(m, n)$. LoRA reduces the number of trainable parameters from $m \times n$ to

*Correspondence to: Jian Liang (liangjian92@gmail.com)

$r \times (m+n)$, thereby decreasing the cost of fine-tuning. However, despite its efficiency, LoRA's fine-tuning performance often falls short compared to full fine-tuning [Hu et al., 2022, Liu et al., 2024, Ding et al., 2023].

In this paper, we propose a novel PEFT method, LoRA-Pro, aimed at bridging the gap between LoRA and full fine-tuning. While LoRA employs low-rank approximation by re-parametrizing weight changes as the product of two low-rank matrices, it falls short in approximating the optimization process of full fine-tuning. To measure their discrepancy in the optimization process, we propose a novel concept, "Equivalent Gradient", for LoRA optimization. Equivalent gradient characterizes the gradient of the original matrix after low-rank approximation (despite it not being directly trainable), is composed of gradients from matrices A and B. Thus, during LoRA fine-tuning, our goal is not only to approximate the matrix with low-rank matrices but also to minimize the difference between the equivalent gradient and the gradient from full fine-tuning during the gradient descent process. This is achieved by selecting appropriate gradients for matrices A and B, ensuring a more accurate and effective fine-tuning process. To achieve this, we formulate it as an optimization problem. We then derive theoretical solutions for the problem, presenting optimal gradients for updating matrices A and B. These solutions ensure that the equivalent gradient closely match the optimization dynamics of full fine-tuning. By doing so, we enhance the effectiveness LoRA, bridging the gap between LoRA and full fine-tuning.

Our main contributions are summarized as follows:

- We identify that LoRA approximates low-rank matrices but neglects to approximate the optimization process of full parameter fine-tuning. This shortcoming is one of the reasons for the performance gap between LoRA and full fine-tuning.
- We introduce the concept of Equivalent Gradient, which allows us to quantify the discrepancy in the optimization process between LoRA and full fine-tuning. By minimizing this discrepancy, we derive the optimal closed-form updated solutions for LoRA.
- Extensive experiments on natural language processing tasks validate the effectiveness of our method.

2 Related Work

Parameter-Efficient Fine-Tuning. Given the huge size of foundation models, recent research has focused on developing parameter-efficient fine-tuning methods [Hu et al., 2022, Liu et al., 2024, Ding et al., 2023, Houlsby et al., 2019, Liu et al., 2023, Lester et al., 2021]. These methods aim to reduce the cost of fine-tuning by adjusting only a small portion of the model's parameters. Generally, these methods fall into two main categories. The first category is adapter tuning [Houlsby et al., 2019, Sung et al., 2022, He et al., 2021, Zhang et al., 2024, Bapna and Firat, 2019, Hu et al., 2022], which involves inserting small neural network modules, called adapters, into specific layers of the model. During fine-tuning, we keep the model frozen and only fine-tune the lightweight adapter modules, significantly reducing the memory footprint for fine-tuning. The second category is prompt tuning [Lester et al., 2021, Zhou et al., 2022, Li and Liang, 2021, Liu et al., 2022]. Prompt tuning adapts the models to specific tasks by adding specially designed prompts or learnable tokens to the input data, rather than directly modifying the internal parameters of foundation models. In this paper, we focus on LoRA [Hu et al., 2022], a prominent method within the realm of adapter tuning.

Low Rank Adaptation. Low-rank adaptation, initially referred to as LoRA [Hu et al., 2022], has evolved into a broad category encompassing parameter-efficient fine-tuning methods based on low-rank approximations [Hu et al., 2022, Liu et al., 2024, Hayou et al., 2024, Kalajdzievski, 2023, Zhang et al., 2023, Kopiczko et al., 2024, Hyeon-Woo et al., 2022, Zhang and Pilanci, 2024, Wang et al., 2024, Zhao et al., 2024]. LoRA [Hu et al., 2022] assumes that the changes in the weights of pre-trained models exhibit a low-rank structure. Consequently, it re-parameterizes these changes as the product of low-rank matrices, thereby reducing the cost associated with fine-tuning.

Several variants of LoRA have been proposed to address different aspects of this approach. For example, DoRA [Liu et al., 2024] improves LoRA [Hu et al., 2022] by incorporating a learnable magnitude vector

to re-scale the normalized product of low-rank matrices. Another variant, rsLoRA Kalajdzievski [2023], introduces a new scaling factor to stabilize training in high-rank scenarios. LoRA+[Hayou et al., 2024] improves upon LoRA by applying different learning rates to the two low-rank matrices. Additionally, Galore [Zhao et al., 2024] employs SVD to project the gradients of full parameter training into a low-rank space, thereby reducing the memory footprint during pre-training and fine-tuning.

3 Method

In this section, we begin by revisiting LoRA [Hu et al., 2022] in Section 3.1. Following this, we conduct a comparison between LoRA and full fine-tuning from an optimization perspective in Section 3.2. Finally, in Section 3.3, we point out that LoRA falls short in approximating full fine-tuning during the optimization process, and we introduce LoRA-Pro as a solution to bridge this performance gap.

3.1 Revisit Low Rank Adaptation

First of all, let's dive back into Low-Rank Adaptation (LoRA) [Hu et al., 2022]. LoRA's core idea revolves around recognizing the low-rank structure of the change matrix ΔW in the standard fine-tuning process. This insight allows LoRA [Hu et al., 2022] to re-parameterize the change matrix into the product of two low-rank matrices,

$$W = W_0 + \Delta W = W_0 + sBA. \tag{1}$$

Here, $W_0 \in \mathbb{R}^{m\times n}$ represents the pre-trained weight matrix, $B \in \mathbb{R}^{m\times r}$ and $A \in \mathbb{R}^{r\times n}$ are the low-rank matrices, and s is a scaling factor. For LoRA [Hu et al., 2022], $s = \frac{\alpha}{r}$, while for rsLoRA [Kalajdzievski, 2023], $s = \frac{\alpha}{\sqrt{r}}$. Here, α is the hyper-parameter and $r \ll min(m,n)$ denotes the rank. Consequently, LoRA significantly reduces the number of fine-tuning parameters from $m \times n$ to $r \times (m+n)$.

3.2 LoRA v.s. Full Fine-tuning

Despite widespread applications across various domains, LoRA's performance still falls short when compared to full fine-tuning. In this part, we review and compare LoRA and full fine-tuning in the optimization process. In full fine-tuning, we utilize differential to analyze the relationship between changes in the loss and changes in the weights:

$$\mathrm{d}L = \langle \frac{\partial L}{\partial W}, \mathrm{d}W \rangle_F, \tag{2}$$

where $\mathrm{d}L$ and $\mathrm{d}W$ denotes the changes of the parameter W and the loss L, and $\|\cdot\|_F$ is the Frobenius norm. To minimize the loss function, we typically set $\mathrm{d}W = -\frac{\partial L}{\partial W} \triangleq -g$ (omitting the learning rate for simplicity), which results in $\mathrm{d}L = -\|\frac{\partial L}{\partial W}\|_F^2 \le 0$.

In LoRA optimization, given that $W = W_0 + sBA$, we compute the differential using the chain rule:

$$\begin{aligned}
\mathrm{d}L &= \langle \frac{\partial L}{\partial W}, \mathrm{d}W \rangle_F \\
&= \langle \frac{\partial L}{\partial W}, \frac{\partial W}{\partial A}^T \mathrm{d}A + \frac{\partial W}{\partial B}^T \mathrm{d}B \rangle_F \\
&= \langle \frac{\partial L}{\partial W}\frac{\partial W}{\partial A}, \mathrm{d}A \rangle_F + \langle \frac{\partial L}{\partial W}\frac{\partial W}{\partial B}, \mathrm{d}B \rangle_F \\
&= \langle \frac{\partial L}{\partial A}, \mathrm{d}A \rangle_F + \langle \frac{\partial L}{\partial B}, \mathrm{d}B \rangle_F.
\end{aligned} \tag{3}$$

Similarly, LoRA sets $\mathrm{d}A = -\frac{\partial L}{\partial A} \triangleq -g^A_{lora}$ and $\mathrm{d}B = -\frac{\partial L}{\partial B} \triangleq -g^B_{lora}$, and thus $\mathrm{d}L = -\|\frac{\partial L}{\partial A}\|_F^2 - \|\frac{\partial L}{\partial B}\|_F^2 \le 0$. Moreover, employing the chain rule, we derive:

$$g^A_{lora} = \frac{\partial L}{\partial W}\frac{\partial W}{\partial A} = sB^T g, \qquad g^B_{lora} = \frac{\partial L}{\partial W}\frac{\partial W}{\partial B} = sgA^T. \tag{4}$$

3.3 Low-Rank Adaptation with Equivalent Gradient

Definition 3.1 (Equivalent Gradient)

In the context of LoRA optimization, we define the equivalent gradient as,

$$\tilde{g} \triangleq \frac{\partial W}{\partial A}^T g^A + \frac{\partial W}{\partial B}^T g^B = sBg^A + sg^B A, \tag{5}$$

where s is the scaling factor, and g^A and g^B are gradients with respect to A and B, respectively.

In this section, **Equivalent Gradient.** From Equation (3), we can see that changes in matrices A and B are inherently linked to changes in matrix W through the chain rule:

$$\mathrm{d}W = \frac{\partial W}{\partial A}^T \mathrm{d}A + \frac{\partial W}{\partial B}^T \mathrm{d}B = -(sBg^A_{lora} + sg^B_{lora}A). \tag{6}$$

In comparison to full fine-tuning, this is equivalent to updating W using the gradient $\tilde{g} = sBg^A_{lora} + sg^B_{lora}A$. This critical relationship has been neglected in the LoRA optimization process. Hence, we hypothesize that by carefully adjusting the gradients of matrices A and B in such a way that $\tilde{g}$ under LoRA closely approximates the gradient g from full fine-tuning, we can effectively bridge the gap between LoRA and full fine-tuning.

Based on this relationship, we define the concept of equivalent gradient in Definition 1. Equivalent gradient describes the gradient of the matrix W following low-rank adaptation, despite W not being a trainable parameter. To narrow the performance gap, our goal is to carefully select suitable g^A and g^B to minimize the distance between the equivalent gradient $\tilde{g}$ and the gradient under full fine-tuning g. Hence, our objective is:

$$\begin{aligned} &\min_{g^A, g^B} \|\tilde{g} - g\|_F^2 \\ \text{s.t.} \quad &\tilde{g} = sBg^A + sg^B A, \\ &\mathrm{d}L \leq 0. \end{aligned} \tag{7}$$

Theorem 3.1

Assume matrices $B \in \mathbb{R}^{m\times r}, A \in \mathbb{R}^{r\times n}$ are both full rank. For the objective $\min_{g^A, g^B} \|\tilde{g} - g\|_F^2$, the solutions are given by:

$$g^A = \frac{1}{s}(B^TB)^{-1}B^Tg + XA = \frac{1}{s^2}(B^TB)^{-1}g^A_{lora} + XA \tag{8}$$

$$g^B = \frac{1}{s}[I - B(B^TB)^{-1}B^T]gA^T(AA^T)^{-1} - BX = \frac{1}{s^2}[I - B(B^TB)^{-1}B^T]g^B_{lora}(AA^T)^{-1} - BX. \tag{9}$$

Here, $X \in \mathbb{R}^{r\times r}$ represents an arbitrary matrix.

Closed-form Solution. Fortunately, Equation (7) admits a closed-form solution. According to Theorem 3.1, we obtain the optimal gradients for matrices A and B, ensuring that the equivalent gradient achieves the best approximation to the full fine-tuning gradient. Moreover, we observe that g^A and g^B can be expressed as g^A_{lora} and g^B_{lora}, respectively, indicating that we do not explicitly possess the full fine-tuning gradient g. Therefore, our approach involves back-propagating in standard LoRA and adjusting the gradients of matrices A and B using the closed-form solution outlined in Theorem 3.1.

Theorem 3.2

When updating matrices A and B using the closed-form solution from Theorem 3.1, we proceed as follows:

$$A \leftarrow A - \gamma g^A \tag{10}$$

$$B \leftarrow B - \gamma g^B, \tag{11}$$

where $\gamma \geq 0$ denotes the learning rate. Our method ensures a decrease in the loss, akin to the standard gradient descent algorithm, expressed by:

$$\mathrm{d}L = -\gamma\{\langle g^A_{lora}, \frac{1}{s^2}(B^TB)^{-1}g^A_{lora}\rangle_F + \langle g^B_{lora}, \frac{1}{s^2}[I - B(B^TB)^{-1}B^T]g^B_{lora}(AA^T)^{-1}\rangle_F\} \leq 0 \tag{12}$$

Although Theorem 3.1 provides a closed-form solution to the optimization problem $\min_{g^A,g^B} \|\tilde{g} - g\|_F^2$, this does not necessarily mean that updating matrices A and B with this solution will decrease the loss. To address this, we have Theorem 3.2, which guarantees a decrease in the loss during the optimization process. This theorem indicates that the change in loss, $\mathrm{d}L$, can be expressed as a negative scalar multiplied by the sum of two positive definite quadratic forms. This relationship ensures that $\mathrm{d}L \leq 0$ during the update process, thus consistently driving the optimization process towards a lower loss.

Theorem 3.3

Consider the optimization problem,

$$\min_X \|g^A - g^A_{lora}\|_F^2 + \|g^B - g^B_{lora}\|_F^2, \tag{13}$$

where g^A and g^B are the optimal solutions as stated in Theorem 3.1. The optimal X can be determined by solving the Sylvester equation:

$$B^TBX + XAA^T = -\frac{1}{s^2}(B^TB)^{-1}g^A_{lora}A^T, \tag{14}$$

which has a unique solution X provided that B^TB and $-AA^T$ do not have any shared eigenvalues.

Selection of X. Although the equivalent gradient itself is not directly related to the matrix X, the presence of X plays a significant role in the updates of matrices A and B. We select an appropriate X such that g^A and g^B remain close to g^A_{lora} and g^B_{lora} respectively. Consequently, we minimize their Frobenius norm, as demonstrated in Equation (41). In practical terms, B^TB and AA^T do not share common eigenvalues. Therefore, according to Theorem 3.3, we can determine a unique optimal X for updating matrices A and B.

4 Experimental Results

In this section, we evaluate our LoRA-Pro method across various natural language understanding datasets. To provide a comprehensive comparison, we include several baseline methods: 1) full fine-tuning and the standard LoRA [Hu et al., 2022]. 2) LoRA variants maintaining the original structure, such as rsLoRA [Kalajdzievski, 2023], LoRA+ [Hayou et al., 2024], PiSSA [Meng et al., 2024], 3) oRA variants with modified structures, including DoRA [Liu et al., 2024] and AdaLoRA [Zhang et al., 2023].

The results are shown in Table 1. We fine-tune the T5-base model [Raffel et al., 2020] with the baseline methods on a subset of GLUE datasets. From Table 1, we observe that LoRA-Pro achieves the highest scores on 3 out of 5 datasets and the highest average score across all 5 datasets. Moreover, on average over 5

datasets, LoRA-Pro suppass standard LoRA [Hu et al., 2022] with a margin of 6.72. These results validate the effectiveness of our methods.

Table 1: Results on fine-tuning T5-base with Full Fine-tuning and LoRA variants on a subset of GLUE datasets.

Method	**MNLI**	**SST2**	**CoLA**	**QNLI**	**MRPC**	**Avg.**
Full FT	86.33±0.00	**94.75±0.21**	80.70±0.24	**93.19±0.22**	84.56±0.73	87.91
LoRA	85.30±0.04	94.04±0.11	69.35±0.05	92.96±0.09	68.38±0.01	82.08
PiSSA	85.75±0.07	94.07±0.06	74.27±0.39	93.15±0.14	76.31±0.51	84.71
rsLoRA	85.73±0.10	94.19±0.23	72.32±1.12	93.12±0.09	52.86±2.27	79.64
LoRA+	85.81±0.09	93.85±0.24	77.53±0.20	93.14±0.03	74.43±1.39	84.95
DoRA	85.67±0.09	94.04±0.53	72.04±0.94	93.04±0.06	68.08±0.51	82.57
AdaLoRA	85.45±0.11	93.69±0.20	69.16±0.24	91.66±0.05	68.14±0.28	81.62
LoRA-GA	85.70±0.09	94.11±0.18	80.57±0.20	93.18±0.06	85.29±0.24	87.77
LoRA-Pro	**86.92±0.08**	94.46±0.24	**82.25±1.01**	92.89±0.12	**87.50±0.65**	**88.80**

5 Conclusion

In this paper, we introduce LoRA-Pro, a novel approach designed to bridge the performance gap between LoRA and full fine-tuning. To bridge the performance gap, we introduce the concept of Equivalent Gradient, which allows us to quantify the difference in the optimization process between LoRA and full fine-tuning. By minimizing this discrepancy, we derive the optimal closed-form updated solutions for LoRA. Moreover, we prove that the solutions guarantee the loss decease during optimization. These solutions not only apply a low-rank approximation to the fine-tuning matrix but also maintain consistency with the optimization of full fine-tuning, enabling more effective fine-tuning. Finally, we validate the effectiveness of our method through extensive experiments on natural language processing tasks.

References

J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

A. Aghajanyan, S. Gupta, and L. Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *ACL-IJCNLP*, 2021.

A. Bapna and O. Firat. Simple, scalable adaptation for neural machine translation. In *EMNLP-IJCNLP*, 2019.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, S. Hu, Y. Chen, C.-M. Chan, W. Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, 2023.

S. Hayou, N. Ghosh, and B. Yu. Lora+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*, 2024.

R. He, L. Liu, H. Ye, Q. Tan, B. Ding, L. Cheng, J. Low, L. Bing, and L. Si. On the effectiveness of adapter-based tuning for pretrained language model adaptation. In *ACL-IJCNLP*, 2021.

J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, et al. Training compute-optimal large language models. In *NeurIPS*, 2022.

N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for nlp. In *ICML*, 2019.

E. J. Hu, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, et al. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022.

N. Hyeon-Woo, M. Ye-Bin, and T.-H. Oh. Fedpara: Low-rank hadamard product for communication-efficient federated learning. In *ICLR*, 2022.

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

J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. In *ICCV*, 2023.

D. J. Kopiczko, T. Blankevoort, and Y. M. Asano. Vera: Vector-based random matrix adaptation. In *ICLR*, 2024.

B. Lester, R. Al-Rfou, and N. Constant. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, 2021.

C. Li, H. Farkhoor, R. Liu, and J. Yosinski. Measuring the intrinsic dimension of objective landscapes. In *ICLR*, 2018.

X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL-IJCNLP*, 2021.

P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

S.-y. Liu, C.-Y. Wang, H. Yin, P. Molchanov, Y.-C. F. Wang, K.-T. Cheng, and M.-H. Chen. Dora: Weight-decomposed low-rank adaptation. In *ICML*, 2024.

X. Liu, K. Ji, Y. Fu, W. Tam, Z. Du, Z. Yang, and J. Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *ACL*, 2022.

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

F. Meng, Z. Wang, and M. Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. *arXiv preprint arXiv:2404.02948*, 2024.

A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. 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.

R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

Y.-L. Sung, J. Cho, and M. Bansal. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *CVPR*, 2022.

I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

S. Wang, L. Yu, and J. Li. Lora-ga: Low-rank adaptation with gradient approximation. *arXiv preprint arXiv:2407.05000*, 2024.

F. Zhang and M. Pilanci. Riemannian preconditioned lora for fine-tuning foundation models. In *ICML*, 2024.

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

R. Zhang, J. Han, C. Liu, A. Zhou, P. Lu, Y. Qiao, H. Li, and P. Gao. Llama-adapter: Efficient fine-tuning of large language models with zero-initialized attention. In *ICLR*, 2024.

J. Zhao, Z. Zhang, B. Chen, Z. Wang, A. Anandkumar, and Y. Tian. Galore: Memory-efficient llm training by gradient low-rank projection. In *ICML*, 2024.

K. Zhou, J. Yang, C. C. Loy, and Z. Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022.

LoRA-Pro: Are Low-Rank Adapters Properly Optimized?

————Appendix————

The structure of Appendix is as follows,

- Appendix A contains the notation usage in our paper.
- Appendix B contains the proofs of the theorems in the main manuscript.
- Appendix C details the optimization algorithm of the proposed methods.

A Notations

In Table 2, we detail the notations utilized in our paper.

Table 2: Description of notations used in the paper.

Notation	Description
s	scaling factor in lora
$B \in \mathbb{R}^{m\times r}$, $A \in \mathbb{R}^{r\times n}$	low rank matrices in LoRA
$g = \frac{\partial L}{\partial W} \in \mathbb{R}^{m\times n}$	gradients of full rank fine-tuning
$g^A_{lora} = \frac{\partial L}{\partial A} = sB^T g \in \mathbb{R}^{r\times n}$	gradients of matrix A in lora
$g^B_{lora} = \frac{\partial L}{\partial B} = sgA^T \in \mathbb{R}^{m\times r}$	gradients of matrix B in lora
$\mathrm{d}L$	differential of the loss function
$\mathrm{d}A$	differential of the matrix A
$\mathrm{d}B$	differential of the matrix B
$\|\cdot\|_F$	Frobenius Norm
$\langle\cdot,\cdot\rangle_F$	Frobenius inner product

B Proof of Theoretical Results

B.1 Proof of Theorem 3.1

Theorem B.1

Assume matrices $B \in \mathbb{R}^{m\times r}, A \in \mathbb{R}^{r\times n}$ are both full rank. For the objective $\min_{g^A,g^B} \|\tilde{g} - g\|_F^2$, the solutions are given by:

$$g^A = \frac{1}{s}(B^TB)^{-1}B^Tg + XA = \frac{1}{s^2}(B^TB)^{-1}g^A_{lora} + XA \tag{15}$$

$$g^B = \frac{1}{s}[I - B(B^TB)^{-1}B^T]gA^T(AA^T)^{-1} - BX = \frac{1}{s^2}[I - B(B^TB)^{-1}B^T]g^B_{lora}(AA^T)^{-1} - BX. \tag{16}$$

Here, $X \in \mathbb{R}^{r\times r}$ represents an arbitrary matrix.

Proof

For simplicity, we denote $L = \|sBg^A + sg^BA - g\|_F^2$. To solve the optimization problem, we need to satisfy the following conditions:

$$\frac{\partial L}{\partial A} = 2sB^T(sBg^A + sg^BA - g) = 0 \tag{17}$$

$$\frac{\partial L}{\partial B} = 2(sBg^A + sg^BA - g)sA^T = 0 \tag{18}$$

Given that matrices A and B are full-rank, AA^T and B^TB are invertible. And from Equation (18), we derive:

$$g^B = \frac{1}{s}gA^T(AA^T)^{-1} - Bg^AA^T(AA^T)^{-1}. \tag{19}$$

Substituting this into Equation (17), we obtain the following linear equation:

$$g^A[I - A^T(AA^T)^{-1}A] = \frac{1}{s}(B^TB)^{-1}B^Tg. \tag{20}$$

Here, we notice that the matrix $P = I - A^T(AA^T)^{-1}A$ is a projection matrix with rank $n - r$. The solution to the linear equation (20) is:

$$g^A = \frac{1}{s}(B^TB)^{-1}B^Tg + XA, \tag{21}$$

where $X \in \mathbb{R}^{r\times r}$ represents an arbitrary matrix. We take the solution (24) into Equation (19), we derive:

$$g^B = \frac{1}{s}[I - B(B^TB)^{-1}B^T]gA^T(AA^T)^{-1} - BX \tag{22}$$

While we have obtained closed-form solutions for g^A and g^B, these solutions explicitly depend on the gradient of the matrix W, i.e., g, which is undesirable since g is unknown during LoRA optimization. Fortunately, the solutions can be transformed into the forms of the gradients of standard LoRA, where the gradients are:

$$g^A_{lora} = sB^Tg, \quad g^B_{lora} = sgA^T. \tag{23}$$

Therefore, the solutions to the optimization problem can be written as:

$$g^A = \frac{1}{s^2}(B^TB)^{-1}g^A_{lora} + XA, \tag{24}$$

$$g^B = \frac{1}{s^2}[I - B(B^TB)^{-1}B^T]g^B_{lora}(AA^T)^{-1} - BX. \tag{25}$$

In our method, we perform the standard forward and backward passes of LoRA, then adjust the gradients of A and B using Solutions (24) and (25), and subsequently update them.

B.2 Proof of Theorem 3.2

Theorem B.2

When updating matrices A and B using the closed-form solution from Theorem 3.1, we proceed as follows:

$$A \leftarrow A - \gamma g^A, \tag{26}$$

$$B \leftarrow B - \gamma g^B, \tag{27}$$

where $\gamma \geq 0$ denotes the learning rate. Our method ensures a decrease in the loss, akin to the standard gradient descent algorithm, expressed by:

$$\mathrm{d}L = -\gamma\{\langle g^A_{lora}, \frac{1}{s^2}(B^TB)^{-1}g^A_{lora}\rangle_F + \langle g^B_{lora}, \frac{1}{s^2}[I - B(B^TB)^{-1}B^T]g^B_{lora}(AA^T)^{-1}\rangle_F\} \leq 0 \tag{28}$$

Proof (Part 1)

In summary, the proof of Theorem 3.2 is divided into two distinct parts. To begin with, we demonstrate that $\mathrm{d}L$ can be expressed in the following form:

$$\mathrm{d}L = -\gamma\{\langle g^A_{lora}, \frac{1}{s^2}(B^TB)^{-1}g^A_{lora}\rangle_F + \langle g^B_{lora}, \frac{1}{s^2}[I - B(B^TB)^{-1}B^T]g^B_{lora}(AA^T)^{-1}\rangle_F\}. \tag{29}$$

In the second part, we prove that this expression for $\mathrm{d}L$ is always less than or equal to zero: $\mathrm{d}L \leq 0$. Therefore, in this part, we first prove Equation (29). During the optimization process, the differential change in the loss function, $\mathrm{d}L$, can be expressed in terms of the differentials $\mathrm{d}A$ and $\mathrm{d}B$ as follows:

$$\mathrm{d}L = \langle \frac{\partial L}{\partial A}, \mathrm{d}A\rangle_F + \langle \frac{\partial L}{\partial B}, \mathrm{d}B\rangle_F. \tag{30}$$

From Equation (26) and (27), we can derive that:

$$\mathrm{d}A = -\gamma g^A, \quad \mathrm{d}B = \gamma g^B. \tag{31}$$

Given that $\frac{\partial L}{\partial A} = g^A_{lora}$ and $\frac{\partial L}{\partial B} = g^B_{lora}$, it follows that:

$$\begin{aligned}
\mathrm{d}L &= -\gamma(\langle g^A_{lora}, g^A\rangle_F + \langle g^B_{lora}, g^B\rangle_F) \\
&= -\gamma(\langle g^A_{lora}, \frac{1}{s^2}(B^TB)^{-1}g^A_{lora}\rangle_F + \langle g^B_{lora}, \frac{1}{s^2}[I - B(B^TB)^{-1}B^T]g^B_{lora}(AA^T)^{-1}\rangle_F \\
&+ \langle g^A_{lora}, XA\rangle_F - \langle g^B_{lora}, BX\rangle_F).
\end{aligned} \tag{32}$$

And we have the following equation:

$$\begin{aligned}
&\langle g^A_{lora}, XA\rangle_F - \langle g^B_{lora}, BX\rangle_F \\
=&\langle g^A_{lora}A^T, X\rangle_F - \langle B^Tg^B_{lora}, X\rangle_F \\
=&\langle g^A_{lora}A^T - B^Tg^B_{lora}, X\rangle_F \\
=&\langle (sB^Tg)A^T - B^T(sgA^T), X\rangle_F \\
=&0.
\end{aligned} \tag{33}$$

Therefore, we have:

$$\mathrm{d}L = -\gamma\{\langle g^A_{lora}, \frac{1}{s^2}(B^TB)^{-1}g^A_{lora}\rangle_F + \langle g^B_{lora}, \frac{1}{s^2}[I - B(B^TB)^{-1}B^T]g^B_{lora}(AA^T)^{-1}\rangle_F\}. \tag{34}$$

Proof (Part 2)

In this part, we aim to prove $\mathrm{d}L \leq 0$. Given that the learning rate $\gamma > 0$, it suffices to show the following inequalities:

$$\langle g^A_{lora}, \frac{1}{s^2}(B^TB)^{-1} g^A_{lora} \rangle_F \geq 0, \tag{35}$$

$$\langle g^B_{lora}, \frac{1}{s^2}[I - B(B^TB)^{-1}B^T] g^B_{lora}(AA^T)^{-1} \rangle_F \geq 0. \tag{36}$$

By proving these inequalities, we can establish that $\mathrm{d}L \leq 0$ as derived from Equation (29).
① Proof of $\langle g^A_{lora}, \frac{1}{s^2}(B^TB)^{-1} g^A_{lora} \rangle_F \geq 0$.
To begin with, we need to show that $(B^TB)^{-1}$ is positive definite. To establish this, it is sufficient to show that B^TB is positive definite, as the inverse of a positive definite matrix is also positive definite. To achieve this, consider any non-zero vector x, and noting that B is full-rank, we have,

$$\langle x, B^TBx \rangle = \langle Bx, Bx \rangle = \|Bx\|^2 > 0. \tag{37}$$

This shows that B^TB is positive definite. Consequently, $(B^TB)^{-1}$ is positive definite as well. Since $(B^TB)^{-1}$ is positive definite, and thus we can apply Cholesky decomposition, and $(B^TB)^{-1} = UU^T$. With this, we have,

$$\begin{aligned} \langle g^A_{lora}, \frac{1}{s^2}(B^TB)^{-1} g^A_{lora} \rangle_F &= \frac{1}{s^2}\langle g^A_{lora}, UU^T g^A_{lora} \rangle_F \\ &= \frac{1}{s^2}\langle U^T g^A_{lora}, U^T g^A_{lora} \rangle_F \\ &= \frac{1}{s^2}\|U^T g^A_{lora}\|^2_F \geq 0 \end{aligned} \tag{38}$$

② Proof of $\langle g^B_{lora}, \frac{1}{s^2}[I - B(B^TB)^{-1}B^T] g^B_{lora}(AA^T)^{-1} \rangle_F \geq 0$.
Similarly, we can prove that matrix $(AA^T)^{-1}$ is positive-definite. By employing Cholesky decomposition, we express $(AA^T)^{-1} = UU^T$, where U is a lower-triangle matrix. Subsequently, we define $P = I - B(B^TB)^{-1}B^T$. It can be shown that $P^2 = P$, indicating that P is a projection matrix. Consequently, the eigenvalues of P are either 0 or 1, which implies that P is positive semi-definite. Utilizing the Cholesky decomposition, we derive that $P = VV^T$, where V is a lower-triangle matrix. Finally, we have:

$$\begin{aligned} \langle g^B_{lora}, \frac{1}{s^2}[I - B(B^TB)^{-1}B^T] g^B_{lora}(AA^T)^{-1} \rangle_F &= \frac{1}{s^2}\langle g^B_{lora}, VV^T g^B_{lora} UU^t \rangle_F \\ &= \frac{1}{s^2}\langle V^T g^B_{lora} U, V^T g^B_{lora} U \rangle_F \\ &= \frac{1}{s^2}\|V^T g^B_{lora} U\|^2_F \geq 0 \end{aligned} \tag{39}$$

In summary, based on the above proofs, we have demonstrated that:

$$\mathrm{d}L = -\gamma\{\langle g^A_{lora}, \frac{1}{s^2}(B^TB)^{-1} g^A_{lora} \rangle_F + \langle g^B_{lora}, \frac{1}{s^2}[I - B(B^TB)^{-1}B^T] g^B_{lora}(AA^T)^{-1} \rangle_F\} \leq 0 \tag{40}$$

B.3 Proof of Theorem 3.3

Theorem B.3

Consider the optimization problem,

$$\min_X \|g^A - g^A_{lora}\|_F^2 + \|g^B - g^B_{lora}\|_F^2, \tag{41}$$

where g^A and g^B are the optimal solutions as stated in Theorem 3.1. The optimal X can be determined by solving the Sylvester equation:

$$B^T BX + XAA^T = -\frac{1}{s^2}(B^T B)^{-1} g^A_{lora} A^T, \tag{42}$$

which has a unique solution X provided that $B^T B$ and $-AA^T$ do not have any shared eigenvalues.

Proof

For simplicity, we denote $L = \|g^A - g^A_{lora}\|_F^2 + \|g^B - g^B_{lora}\|_F^2$. To solve the optimization problem, we need to satisfy the following conditions:

$$\frac{\partial L}{\partial X} = 0. \tag{43}$$

Since g^A and g^B are solutions in Theorem 3.1 and $g^A_{lora} = sB^T g$ and $g^B_{lora} = sgA^T$, we obtain that:

$$\begin{aligned} & 2(g^A - g^A_{lora})A^T - 2B^T(g^B - g^B_{lora}) = 0, \\ \Rightarrow \quad & g^A A^T - B^T g^B = g^A_{lora} A^T - B^T g^B_{lora}, \\ \Rightarrow \quad & B^T BX + XAA^T = -\frac{1}{s^2}(B^T B)^{-1} g^A_{lora} A^T, \end{aligned} \tag{44}$$

which is a Sylvester equation. This equation has a unique solution for X if and only if $B^T B$ and $-AA^T$ have no shared eigenvalues.

C Optimization Algorithms

In this section, we present the pseudo-codes for implementing our LoRA-Pro method using the SGD [Sutskever et al., 2013] and AdamW [Loshchilov and Hutter, 2019] optimizers. These are detailed in Algorithm 1 and Algorithm 2, respectively.

In the standard SGD algorithm, as illustrated in Algorithm 1, all we need to do is adjusting the gradients of matrices A and B with the solutions in Theorem 3.1.

In AdamW optimizer, the implementation becomes more complex. Several modifications are necessary. Firstly, in order to mimic full fine-tuning, after adjusting the gradients of matrices A and B, we need to compute the equivalent gradient,

$$\tilde{g} = sg^B A + sBg^A. \tag{45}$$

Subsequently, we calculate the first and second moments of this equivalent gradient to derive the corresponding AdamW gradient, $\tilde{g}^{AdamW}$. Secondly, we determine the gradients with respect to matrices A and B as follows:

$$\tilde{g}^A = sB^T \tilde{g}^{AdamW}, \quad \tilde{g}^B = s\tilde{g}^{AdamW} A^T. \tag{46}$$

Thirdly, the weight decay process must be adjusted. In line with full fine-tuning, the weight decay is given by:

$$W \leftarrow (1 - \gamma\lambda)(W_0 + sBA). \tag{47}$$

This can be decomposed into:

$$W_0 \leftarrow (1-\gamma\lambda)W_0, \quad B \leftarrow \sqrt{1-\gamma\lambda}B, \quad A \leftarrow \sqrt{1-\gamma\lambda}A \tag{48}$$

Algorithm 1 LoRA-Pro with SGD optimizer

Require: Given initial learning rate γ, scaling factor s.

Initialize time step $t \leftarrow 0$, low-rank matrices $A_0 \in \mathbb{R}^{r\times n}$ and $B_0 \in \mathbb{R}^{m\times r}$
repeat
  $t \leftarrow t+1$
  $g^A_{lora}, g^B_{lora} \leftarrow \text{SelectBatch}(A_{t-1}, B_{t-1})$ ▷ *Select batch and return the corresponding gradients*
  $A, B \leftarrow A_{t-1}, B_{t-1}$ ▷ *Obtain the low-rank matrices A and B*
  $X \leftarrow \text{SolveSylvester}(B^TBX + XAA^T = -\frac{1}{s^2}(B^TB)^{-1}g^A_{lora}A^T)$ ▷ *Compute X by solving the sylvester equation*
  $g^A = \frac{1}{s^2}(B^TB)^{-1}g^A_{lora} + XA$ ▷ *Adjust the gradients of LoRA with Theorem 3.1*
  $g^B = \frac{1}{s^2}[I - B(B^TB)^{-1}B^T]g^B_{lora}(AA^T)^{-1} - BX$
  $A_t \leftarrow A_{t-1} - \gamma g^A$
  $B_t \leftarrow B_{t-1} - \gamma g^B$
until *stopping criterion is met*
return optimized parameters A_t and B_t

Algorithm 2 LoRA-Pro with AdamW optimizer

Require: Given initial learning rate γ, scaling factor s, original weight matrix $W_0 \in \mathbb{R}^{m\times n}$, and $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, \lambda \in \mathbb{R}$

Initialize time step $t \leftarrow 0$, low-rank matrices $A_0 \in \mathbb{R}^{r\times n}$ and $B_0 \in \mathbb{R}^{m\times r}$, first momentum $m_0 \in \mathbb{R}^{m\times n}$, second momentum $v_t \in \mathbb{R}^{m\times n}$

repeat

$t \leftarrow t+1$

$g^A_{lora}, g^B_{lora} \leftarrow \text{SelectBatch}(A_{t-1}, B_{t-1})$ ▷ *Select batch and return the corresponding gradients*

$A, B \leftarrow A_{t-1}, B_{t-1}$ ▷ *Obtain the low-rank matrices A and B*

$X \leftarrow \text{SolveSylvester}(B^TBX + XAA^T = -\frac{1}{s^2}(B^TB)^{-1}g^A_{lora}A^T)$ ▷ *Compute X by solving the sylvester equation*

$g^A = \frac{1}{s^2}(B^TB)^{-1}g^A_{lora} + XA$ ▷ *Adjust the gradients of LoRA with Theorem 3.1*

$g^B = \frac{1}{s^2}[I - B(B^TB)^{-1}B^T]g^B_{lora}(AA^T)^{-1} - BX$

$\tilde{g} \leftarrow sg^BA + sBg^A$ ▷ *Compute equivalent gradient*

$m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1)\tilde{g}$

$v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2)\tilde{g}^2$

$\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$

$\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$

$\tilde{g}^{AdamW} \leftarrow \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$

$\tilde{g}^A_{lora} \leftarrow sB^T\tilde{g}^{AdamW}$

$\tilde{g}^B_{lora} \leftarrow s\tilde{g}^{AdamW}A^T$

$X \leftarrow \text{SolveSylvester}(B^TBX + XAA^T = -\frac{1}{s^2}(B^TB)^{-1}\tilde{g}^A_{lora}A^T)$

$\tilde{g}^A = \frac{1}{s^2}(B^TB)^{-1}\tilde{g}^A_{lora} + XA$ ▷ *Adjust the gradients of LoRA with Theorem 3.1*

$\tilde{g}^B = \frac{1}{s^2}[I - B(B^TB)^{-1}B^T]\tilde{g}^B_{lora}(AA^T)^{-1} - BX$

$A \leftarrow \sqrt{1-\gamma\lambda}A$ ▷ *Weight Decay*

$B \leftarrow \sqrt{1-\gamma\lambda}B$

$W_0 \leftarrow (1-\gamma\lambda)W_0$

$A_t \leftarrow A_{t-1} - \gamma\tilde{g}^A$

$B_t \leftarrow B_{t-1} - \gamma\tilde{g}^B$

until *stopping criterion is met*

return optimized parameters A_t and B_t