TLoRA: Task-aware Low Rank Adaptation of Large Language Models

Weicheng Lin*, **Yi Zhang***, **Jiawei Dang**, **Liang-Jie Zhang**†
College of Computer Science and Software Engineering, Shenzhen University, China
2410105042@mails.szu.edu.cn, rambo.ai@szu.edu.cn, zhanglj@ieee.org

Abstract

Low-Rank Adaptation (LoRA) has become a widely adopted parameter-efficient fine-tuning method for large language models, with its effectiveness largely influenced by the allocation of ranks and scaling factors, as well as initialization. Existing LoRA variants typically address only one of these factors, often at the cost of increased training complexity or reduced practical efficiency. In this work, we present Task-aware Low-Rank Adaptation (TLoRA), a unified framework that jointly optimizes initialization and resource allocation at the outset of training. TLoRA introduces a data-driven initialization strategy that aligns the LoRA A matrix with task-relevant subspaces by performing singular value decomposition on the product of pre-trained weights and input activation covariance. After this, the A matrix is frozen, and only the B matrix is trained. Furthermore, TLoRA employs a sensitivity-based importance metric to adaptively allocate ranks and scaling factors across layers under a fixed parameter budget. We conduct extensive experiments that demonstrate TLoRA consistently performs excellently across various tasks, including natural language understanding, commonsense reasoning, math reasoning, code generation, and chat generation, while significantly reducing the number of trainable parameters. Our code is available at https://github.com/Rambo-Yi/TLora/tree/main

1 Introduction

Large Language Models (LLMs) exhibit remarkable capabilities across diverse Natural Language Processing (NLP) tasks (Achiam et al., 2023; Ouyang et al., 2022; Guo et al., 2025). However, as model scales expand to billions or even hundreds of billions of parameters, full fine-tuning imposes prohibitive computational and memory overhead (Grattafiori et al., 2024; Hoffmann et al., 2022). To address this, Parameter-Efficient Fine-Tuning (PEFT) has emerged as a solution that achieves performance comparable to full fine-tuning by updating only a minimal number of parameters, significantly reducing the requirements for domain-specific fine-tuning (Houlsby et al., 2019; Liu et al., 2021; Ding et al., 2023).

Among the various PEFT techniques, Low-Rank Adaptation (LoRA) (Hu et al., 2022) stands out as a notable approach. As depicted in Figure 1 (Left), LoRA keeps the pre-trained weights W frozen and approximates the weight update ΔW via two low-rank matrices, A and B. In the standard implementation, A is initialized with a Gaussian distribution, and B is initialized to zero, ensuring that the initial model output remains identical to the pre-trained model. This design significantly reduces the number of trainable parameters and enables the product BA to be merged into W during inference, thereby avoiding extra inference latency. However, while LoRA performs well on simple tasks, recent studies indicate that it struggles to match the performance of full fine-tuning on complex tasks (Shuttleworth et al., 2024; Biderman et al., 2024).

A significant factor influencing the effectiveness of LoRA is the initialization strategy. Fundamentally, matrix A functions as a feature extractor that captures key low-rank information from high-dimensional inputs, while matrix B serves as an output mapper that projects these features into the output space (Zhu et al., 2024). However, the random initialization of standard LoRA often results in a low-rank subspace defined by A that is misaligned with task-relevant directions. This forces the model to consume substantial optimization steps at the beginning of training, primarily to rotate the projection subspace toward task-relevant directions, thereby hindering convergence efficiency. Motivated by this analysis, we

*These authors contributed equally to this work.
†Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 29252–29269

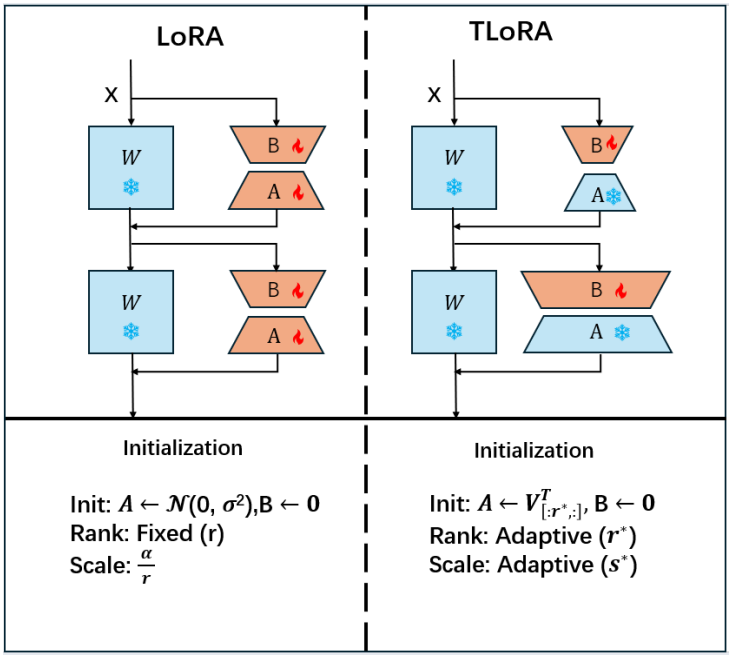

Figure 1: **(Left)** Standard LoRA employs random Gaussian initialization for matrix A and initializes B to zero, using a fixed rank r across all modules. **(Right)** TLoRA utilizes a task-aware initialization strategy where matrix A is initialized using the top-r^* singular vectors of the product of pre-trained weights and input covariance (W_0C) and is subsequently frozen (indicated by the snowflake). Furthermore, TLoRA adopts an adaptive resource allocation mechanism to dynamically determine the optimal rank r^* and scaling factor s^* for each module based on importance scores to ensure effective optimization.

hypothesize that if A can be precisely aligned with task-relevant feature subspaces during the initialization phase, the feature extractor can remain frozen throughout subsequent training, requiring only the optimization of the mapper B to achieve efficient adaptation.

Another significant factor influencing the effectiveness of LoRA is the selection of rank and scaling factors. There are substantial differences in the contributions of different layers to downstream tasks. The uniform rank and scaling strategy of standard LoRA implicitly assumes that all layers contribute equally, which often leads to parameter wastage in non-critical layers while restricting the adaptation capacity of critical layers. Although most existing methods dynamically adjust the rank of each layer during the training process, this inevitably increases training overhead. More importantly, the α/r scaling strategy in standard LoRA actually limits the influence of critical layers when the rank varies. Therefore, we require a mechanism that avoids additional training costs and adaptively allocates resources based on module importance right at the initialization stage.

In light of these challenges, we propose Task-aware Low-Rank Adaptation (TLoRA), as illustrated in Figure 1 (Right). First, to address the initialization alignment issue, we introduce a data-driven initialization method. Specifically, we compute the input activation covariance matrix on a subset of training samples and perform Singular Value Decomposition (SVD) on the product of the pre-trained weight matrix and this activation covariance. The resulting right singular vectors are used to initialize and freeze matrix A; meanwhile, matrix B is initialized to zero and remains the only trainable component. Second, to address the selection of rank and scaling factors, we utilize a sensitivity-based weight metric to assess the importance of each module and allocate ranks and scaling factors accordingly, ensuring that the resource budget is concentrated on the most critical components. Our contributions are summarized as follows:

1. We derive the optimal closed-form solution A^* under the constraint of a frozen projection matrix A. This result provides critical guidance for designing our initialization strategy.

2. We introduce an importance scoring mechanism that adaptively assigns different ranks and scaling factors to each module, enabling more efficient adaptation without increasing the parameter budget.

3. We conduct comprehensive experiments across several challenging tasks, and the results demonstrate that TLoRA significantly outperforms LoRA and its variants on multiple benchmarks, even with an approximately 50% reduction in parameter count.

2 Related Work

2.1 Parameter-Efficient Fine-Tuning (PEFT)

PEFT methods aim to reduce computational and memory costs by fine-tuning only a small subset of model parameters while maintaining performance comparable to full fine-tuning (Ding et al., 2023; Xu et al., 2023b). A variety of PEFT techniques have been proposed in recent years. Adapter-based methods insert small trainable modules between the frozen layers of a pretrained model (Houlsby et al., 2019; Rücklé et al., 2020; Pfeiffer et al., 2020). While significantly reducing the number of trainable parameters, this approach typically incurs additional computational overhead, leading to increased inference latency. Prompt-based methods add trainable vectors to the beginning of the input sequence to enable effective fine-tuning (Li and Liang, 2021; Lester et al., 2021; Liu et al., 2021). Although these methods demonstrate strong performance across a wide range of tasks, the extended input sequence length results in increased computational overhead during inference. LoRA represents a particularly successful branch of PEFT methods (Hu et al., 2022). It achieves effective adaptation by using low-rank decomposition matrices to approximate the weight updates, without introducing any additional inference latency after weight merging. This practical solution has inspired numerous extensions and improvements. Our work focuses on optimizing the initialization, rank, and scaling factor allocation to further unlock the potential of low-rank adaptation.

2.2 LoRA Initialization

LoRA initialization is crucial for convergence speed and final performance. Many studies show that Vanilla LoRA's reliance on random Gaussian initialization limits its ability to match Full Fine-Tuning on complex tasks. Current improvements to initialization primarily fall into two categories.

The first category is weight-driven initialization. PiSSA performs Singular Value Decomposition (SVD) on pre-trained weights and initializes LoRA adapters using the principal singular vectors and values (Meng et al., 2024); in contrast, MiLoRA initializes adapters using the least significant singular vectors and values (Wang et al., 2025); and OLoRA utilizes QR decomposition on pre-trained weights for initialization (Büyükakyüz, 2024).

The second category is data-driven initialization. LoRA-GA aims to mitigate the initialization misalignment by aligning the adapter's initial gradients with those of Full Fine-Tuning (Wang et al., 2024), and CorDA decomposes weights using context-oriented covariance matrices to preserve knowledge-dependent components (Yang et al., 2024). While these methods enhance LoRA's performance across various tasks to some extent, they typically require modifying the pre-trained model weights ($W_{new} = W_0 - A_0B_0$) to ensure that the model's output remains unaffected at the onset of training. This compromise undermines LoRA's advantage of efficient inference, as it requires either consuming significant time to reconstruct the initialization during inference or storing an additional set of weights for subsequent loading.

2.3 Asymmetry and Adaptive Allocation

Recent investigations into the internal mechanisms of LoRA have revealed a significant asymmetry between the two low-rank matrices: matrix A is responsible for extracting features from the input, while B projects these features onto the output (Zhu et al., 2024). This functional discrepancy has motivated a range of targeted optimization strategies. For example, LoRA+ (Hayou et al., 2024) achieves enhanced performance by increasing the learning rate specifically for matrix B. LoRA-FA (Zhang et al., 2023a) demonstrates that keeping A frozen throughout the training process while training only matrix B yields performance comparable to LoRA; this finding not only reduces memory overhead but also empirically suggests the feasibility of constructing a fixed, high-quality feature extractor A.

Furthermore, the selection of rank plays a decisive role in the performance of LoRA. It is widely acknowledged that increasing the rank dimension enhances the model's fitting capability. However, this improvement comes at the cost of linearly increasing GPU memory footprint and computational expense. Consequently, maximizing fine-tuning efficacy within a limited parameter budget has emerged as a focal point of current re-

search. Recent works such as AdaLoRA (Zhang et al., 2023b) utilize SVD to iteratively prune singular values, achieving adaptive budget allocation across different layers and modules. Alternatively, DyLoRA (Valipour et al., 2023) introduces a dynamic training objective that allows a single model to support multiple ranks simultaneously by training on a nested subset of low-rank matrices. However, a common limitation of this class of methods is their reliance on dynamic adjustments during training, which inevitably introduces additional computational overhead and significantly increases the complexity of the training pipeline.

3 Method

3.1 Background: Low-Rank Adaptation (LoRA)

LoRA assumes that the weight updates required to adapt a pretrained model to downstream tasks lie in a low-dimensional subspace (Hu et al., 2022; Aghajanyan et al., 2020; Li et al., 2018). Formally, for a given pretrained weight matrix $W_0 \in \mathbb{R}^{m\times n}$, LoRA approximates the update ΔW as follows:

$$W' = W_0 + \Delta W = W_0 + \frac{\alpha}{r} BA, \qquad (1)$$

where $W' \in \mathbb{R}^{m\times n}$, $A \in \mathbb{R}^{r\times n}$, $B \in \mathbb{R}^{m\times r}$, $r \ll \min(m, n)$, and $\frac{\alpha}{r}$ serves as a scaling factor for the update.

During fine-tuning, W_0 remains fixed, A is initialized with a random Gaussian distribution, and B is zero-initialized, ensuring that the model's outputs remain initially unchanged.

3.2 Task-Aware Initialization

To elucidate the critical role of matrix A during the fine-tuning process, we first analyze the update dynamics of TLoRA subject to the constraint of a frozen A.

According to Eq. 1, the gradient of the loss function $\mathcal{L}$ with respect to B is $\frac{\partial \mathcal{L}}{\partial B} = \frac{\partial \mathcal{L}}{\partial W} A^T$. Therefore, during the training process, the update ΔW of the entire weight matrix can be expressed as:

$$\Delta W = \frac{\alpha}{r} \Delta B A = -\eta \frac{\alpha}{r} \frac{\partial \mathcal{L}}{\partial W} A^T A \qquad (2)$$

where η represents the learning rate. Eq. 2 reveals a key mechanism: the weight update ΔW is strictly constrained within the subspace spanned by the row vectors of A. This mechanism implies that if the row space of A fails to effectively capture task-relevant feature directions, the model will be confined to a suboptimal subspace regardless of the optimization of B, thereby preventing performance from reaching the upper bound of full fine-tuning.

To determine the optimal initialization of A under this constraint, we provide a detailed theoretical derivation in Appendix A. We formulate the problem as minimizing the reconstruction error between the low-rank approximation and the ideal weight update. By solving this optimization problem, the closed-form solution is derived as:

$$A = V_r^T C^{-1/2} \in \mathbb{R}^{r\times n} \qquad (3)$$

where C is the input activation covariance matrix, and V_r represents the top-r right singular vectors derived from the Singular Value Decomposition (SVD) of $W_0 C^{1/2}$. However, directly applying this theoretical solution encounters severe numerical stability challenges. As detailed in Appendix B.1, the activation covariance matrix C of LLMs typically exhibits an ill-conditioned spectrum characterized by a long tail of small eigenvalues and noise. In such scenarios, computing $C^{-1/2}$ necessitates inverting extremely small eigenvalues, which disproportionately amplifies task-irrelevant noise within the tail singular vectors, thereby causing extreme instability in the feature extractor during the initial training phase or even triggering gradient explosion.

To this end, TLoRA employs a robust approximation strategy. Instead of relying on unstable theoretical targets, we directly perform singular value decomposition (SVD) on the product of pretrained weights and the covariance matrix ($W_0 C$), using the top-r right singular vectors to initialize the A matrix.

Although $W_0 C$ and the theoretical target $W_0 C^{1/2}$ differ mathematically, our empirical analysis in Appendix B.2 indicates that they extract nearly identical feature subspaces. Specifically, we observe an average subspace similarity of 0.908 between the singular vectors of the two targets across all layers. Moreover, theoretically, this approximation serves as a beneficial regularization effect. Since C weights feature directions based on variance (λ), while $C^{1/2}$ weights are based on standard deviation ($\sqrt{\lambda}$), using $W_0 C$ imposes stronger penalties on tail noise directions. This effectively suppresses tail noise while preserving dominant task-specific directions validated by high subspace overlap.

Algorithm 1 TLoRA Initialization

Input: Pre-trained model weights $\{W_i\}_{i=1}^{L}$, sample size N, LoRA rank r_{init}, LoRA alpha α
Output: Initialized LoRA modules $\{(A_i, B_i, \alpha_i, r_i)\}_{i=1}^{L}$.
for $i = 1$ to L **do**
  $S_i \leftarrow 0$,$C_i \leftarrow 0$
end for
for $n = 1$ to N **do**
  for $i = 1$ to L **do**
    $S_i \leftarrow S_i + \frac{1}{N}\, avg(|W_i \cdot \nabla_{W_i}\mathcal{L}|)$
    $C_i \leftarrow C_i + \frac{1}{N} X_i^{\top} X_i$.
  end for
end for
$R_{total} \leftarrow L \cdot r_{init}$, $\alpha_{total} \leftarrow L \cdot \alpha$
for $i = 1$ to L **do**
  $r_i \leftarrow \left\lfloor R_{\text{total}} \cdot \frac{\mathcal{S}(W_i)}{\sum_{j=1}^{L} \mathcal{S}(W_j)} \right\rceil$
  $\alpha_i \leftarrow r_i \cdot \frac{\alpha_{total}}{r_{init}} \cdot \frac{S(W_i)}{\sum_{j=1}^{L} S(W_j)}$
  $U, S, V^{\top} \leftarrow \text{SVD}(W_i C_i)$.
  $A_i \leftarrow V^{\top}_{[:r_i,:]}$, $B_i \leftarrow 0$.
end for
return $\{(A_i, B_i, \alpha_i, r_i)\}_{i=1}^{L}$

Based on this analysis, the final initialization scheme is defined as:

$$A = V_r^T \quad B = 0 \tag{4}$$

where V_r^T denotes the top-r right singular vectors of the matrix $W_0 C$. This approximation method effectively captures task-relevant directions while maintaining a high degree of numerical stability.

3.3 Adaptive Rank and Scaling Assignment

Our Task-Aware Initialization resolves the optimization of the projection subspace for an individual module. As derived in Appendix A, for a given rank r, the maximum objective value achievable by the initialization matrix is determined by the sum of its top-r task-specific singular values: $\sum_{i=1}^{r} \sigma_i(M_\Delta)$.

However, when extending our perspective to the entire model comprising L modules, the global optimization objective becomes maximizing the total objective value across the system: $\sum_{n=1}^{L} \sum_{i=1}^{r_n} \sigma_i(M_\Delta^{(n)})$. The singular value spectra of M_Δ vary significantly across different modules, which fundamentally reflects their differing degrees of contribution to downstream tasks. Therefore, under a fixed total rank budget, a uniform rank assignment strategy is mathematically suboptimal. To maximize the global optimization objective, the rank r_n must be dynamically allocated, assigning larger rank capacities to modules with higher task importance. This reveals a tight theoretical coupling: initialization aligns the directions of the subspace, while rank allocation optimizes its capacity. Motivated by this unified global objective, we propose an adaptive allocation strategy that distributes the limited parameter budget to the more critical modules, thereby achieving more efficient model adaptation under the same budget.

To quantify each module's contribution, we employ a sensitivity-based importance metric (Liang et al., 2021; Zhang et al., 2022). The underlying principle of this metric is that the most critical parameters are those with both a large magnitude and a significant impact on the loss. To define the importance score for a given module i, $\mathcal{S}(W_i)$, we first capture the influence of each parameter by calculating the product of its magnitude and gradient $|w \cdot \nabla_w \mathcal{L}|$, and then average these values across all parameters within the module:

$$\mathcal{S}(W_i) = \frac{1}{|W_i|} \sum_{w \in W_i} |w \cdot \nabla_w \mathcal{L}| \tag{5}$$

where $w \in W_i$ denotes each parameter in the weight matrix W_i, $\nabla_w \mathcal{L}$ is the gradient of the loss with respect to w, and $|W_i|$ represents the number of parameters in W_i. Given the total rank budget

R_{total} (sum of distributable ranks), we allocate a specific rank r_i to each module i proportional to its importance score:

$$r_i = \left\lfloor R_{\text{total}} \cdot \frac{\mathcal{S}(W_i)}{\sum_{j=1}^{L} \mathcal{S}(W_j)} \right\rceil \tag{6}$$

Furthermore, to ensure that critical modules assigned a high rank can be effectively trained, it is necessary to adjust the scaling factor strategy. Standard LoRA employs a uniform scaling factor $s = \frac{\alpha}{r}$ across all modules. However, this uniform approach treats all modules indiscriminately, failing to account for disparities in their importance to the task. In our design, we concentrate the valuable rank budget on the critical modules most sensitive to the task. Adhering to the standard scaling of $\frac{\alpha}{r}$ would paradoxically cause these critical high-rank modules to receive smaller scaling factors, thereby inadvertently attenuating the update magnitude of the most critical modules.

Following the assignment of adaptive ranks based on importance scoring, we recalibrate the scaling factor to amplify the contributions of key modules. Specifically, instead of using a fixed α, we dynamically calculate α_i for each module based on the importance score of the module:

$$\alpha_i = r_i \cdot \frac{\alpha_{total}}{r_{init}} \cdot \frac{\mathcal{S}(W_i)}{\sum_{j=1}^{L} \mathcal{S}(W_j)} \tag{7}$$

where α_{total} represents the total scaling budget across all layers, and r_{init} denotes the initial rank (e.g., $r = 8$). This formulation ensures that modules with higher importance scores $\mathcal{S}(W_i)$ are assigned both a larger rank r_i and a correspondingly higher update magnitude α_i, strictly adhering to our resource-concentration design objective. The detailed algorithm for TLoRA is in Algorithm 1.

4 Experiments

We conduct a series of experiments to comprehensively evaluate TLoRA's performance across various scenarios. Initially, we evaluate its Natural Language Understanding (NLU) capabilities using the T5-base (Raffel et al., 2020) model across a subset of GLUE (Wang et al., 2018). Subsequently, to further evaluate TLoRA's performance in natural language generation (NLG), we perform extensive experiments on the LLaMA2-7B model (Touvron et al., 2023), covering commonsense reasoning, math reasoning, code generation, and chat generation. Finally, we conduct an ablation study to validate TLoRA's effectiveness. We execute all experiments on a single NVIDIA A800 GPU. Hyperparameters are detailed in Appendix C.

4.1 Natural Language Understanding

Settings. We evaluate TLoRA and baseline methods on the T5-base model. Our experimental evaluation covers multiple tasks from the GLUE benchmark, including MRPC (Dolan and Brockett, 2005), CoLA (Warstadt et al., 2019), RTE (Bentivogli et al., 2009), SST-2 (Socher et al., 2013), and QNLI (Rajpurkar et al., 2016). We use accuracy as the evaluation metric across all tasks to ensure a consistent comparison.

Results. Table 1 shows the performance of TLoRA and several baselines on five representative tasks in the GLUE benchmark. TLoRA attains the highest scores on three of the five tasks (MRPC, COLA, RTE). While it maintains competitive performance on SST-2 and QNLI, TLoRA's average score (85.96%) outperforms all baselines. These results substantiate TLoRA's effectiveness and versatility across diverse natural language understanding (NLU) tasks.

Method	Trainable Parameters	MRPC	COLA	RTE	SST-2	QNLI	Avg.
FULL	222.90M	87.99	81.30	58.48	93.69	**93.09**	82.91
LoRA	12.97M	85.53	76.03	63.17	94.49	93.00	83.44
AdaLoRA	12.98M	71.81	81.01	54.87	93.69	92.89	78.85
LoRA+	12.97M	66.66	70.75	51.26	93.80	92.73	75.04
DoRA	13.17M	86.51	81.20	59.56	94.26	93.06	82.91
OLoRA	12.97M	87.99	75.83	66.78	90.13	90.48	82.24
PISSA	12.97M	88.72	81.20	69.67	94.03	92.60	85.24
LoRA-GA	12.97M	88.23	82.83	69.31	**94.61**	93.00	85.59
CorDA	12.97M	88.48	81.78	71.11	93.92	92.75	85.60
TLoRA	**5.44M**	**88.97**	**82.93**	**71.48**	93.80	92.62	**85.96**

Table 1: Results of fine-tuning T5-base using TLoRA and baseline method on a subset of GLUE. Bold numbers indicate the best performance achieved on this subtask. For TLoRA, the reported trainable parameters (5.44M) are the average across five tasks due to its adaptive rank mechanism.

4.2 Natural Language Generation

Settings. We conduct extensive experiments on the LLaMA2-7B model across four diverse task categories, each designed to evaluate a distinct capability of large language models: commonsense reasoning, math reasoning, code generation, and chat generation. These tasks serve as comprehensive benchmarks to evaluate the fine-tuning effec-

Method	Trainable Parameters	BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Averge
FULL	6738M	62.17	73.12	74.82	83.35	72.13	72.51	56.99	69.40	70.56
LoRA	79.95M	72.38	85.96	81.52	94.86	86.50	**88.88**	74.06	84.40	83.57
AdaLoRA	79.96M	71.34	83.51	81.62	93.74	84.68	87.58	73.37	83.80	82.46
LoRA+	79.95M	72.14	83.40	80.50	94.17	85.79	87.24	73.37	81.20	82.23
DoRA	81.31M	72.62	85.03	81.52	94.81	85.79	88.29	75.42	85.40	83.61
OLoRA	79.95M	71.34	85.90	80.91	94.15	85.39	87.07	74.57	84.00	82.92
PISSA	79.95M	71.89	85.03	80.96	93.91	85.47	86.78	73.72	**86.40**	83.02
LoRA-GA	79.95M	69.93	84.33	81.26	94.08	85.47	87.07	72.18	84.40	82.34
CorDA	79.95M	67.33	79.86	79.22	91.60	83.10	82.82	68.00	81.40	79.17
TLoRA	**41.68M**	**72.87**	**86.28**	**82.59**	**95.21**	**86.58**	88.59	**76.36**	85.20	**84.21**

Table 2: Evaluation results of commonsense reasoning of LLaMA2-7B on 8 tasks. Bold numbers indicate the best performance achieved on this subtask.

Method	Trainable Parameters	GSM8K	MATH	HumanEval	MBPP	MT-Bench
FULL	6738M	52.31	8.08	23.20	38.60	4.75
LoRA	319.81M	44.80	6.18	20.70	35.70	4.76
AdaLoRA	319.84M	43.20	5.74	20.70	36.00	4.50
LoRA+	319.81M	48.67	6.92	22.60	35.40	4.69
DoRA	321.17M	45.10	5.96	20.70	36.00	4.70
OLoRA	319.81M	52.38	8.22	22.00	38.90	4.99
PISSA	319.81M	53.44	7.40	22.60	38.60	5.00
LoRA-GA	319.81M	**58.15**	8.66	23.20	38.60	4.97
CorDA	319.81M	54.43	8.70	21.30	39.40	5.09
TLoRA	**171.71M**	56.34	**9.08**	**23.50**	**40.20**	**5.17**

Table 3: Results of fine-tuning LLaMA2-7B using TLoRA and baseline method on math reasoning, code generation , and chat generation. Bold numbers indicate the best performance achieved on this subtask. For TLoRA, the reported trainable parameters (171.71M) are the average across three tasks due to its adaptive rank mechanism.

tiveness of TLoRA in comparison with baseline methods.

- **Commonsense Reasoning.** We fine-tune LLaMA2-7B on Commonsense170K (Hu et al., 2023). We use accuracy as an indicator for commonsense reasoning tasks.
- **Math Reasoning.** We fine-tune LLaMA2-7B on a selected 100K subset of MetaMathQA (Yu et al., 2023). We evaluate on two benchmarks: GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021). We measure the accuracy of the final answer.
- **Code Generation.** We fine-tune LLaMA2-7B on a 100K subset of Code-Feedback (Zheng et al., 2024). We evaluate on two benchmarks: HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). Performance is measured using the pass@1 metric, representing the percentage of generated solutions that pass all test cases.
- **Chat Generation.** We fine-tune LLaMA2-7B on a 100K subset of WizardLM-Evol-Instruct (Xu et al., 2023a). We evaluate using MT-Bench (Zheng et al., 2023). Response quality is evaluated using GPT-4 as a judge.

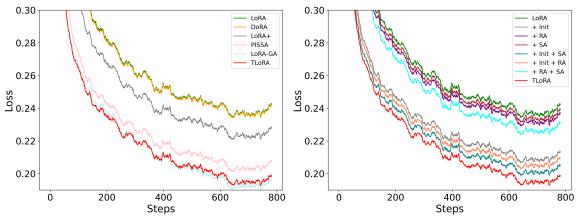

Figure 2: **(Left)** Training loss curves of TLoRA and baseline with 128 ranks on the MetaMathQA dataset. **(Right)** Training loss curves from the ablation study with different settings on the MetaMathQA dataset.

Results. The comprehensive experimental results are shown in Tables 2 and 3. Notably, despite utilizing significantly fewer trainable parameters, TLoRA consistently achieves superior or highly competitive performance across various inference and generative benchmarks. On com-

monsense reasoning benchmarks, TLoRA demonstrates superior performance, achieving the best results on five out of eight subtasks and attaining the highest overall average accuracy (84.21%). In the domain of math reasoning, TLoRA demonstrates robust capabilities. On the MATH math reasoning benchmark, TLoRA achieves an accuracy of 9.08%, outperforming all baselines. For GSM8K, although it trails LoRA-GA marginally, TLoRA still achieves an excellent accuracy rate of 56.34%, significantly surpassing Full Fine-Tuning (+4.03%) and standard LoRA (+11.54%). Similarly, in code generation tasks, which require both algorithmic reasoning and syntactic precision, TLoRA demonstrates clear superiority. It achieves the best performance on both HumanEval (23.50%) and MBPP (40.20%), outperforming all parameter-efficient baselines. Even in the highly complex domain of conversational AI, TLoRA surpasses all baseline methods and maintains optimal performance (5.17). Additionally, Figure 2 (Left) shows that our method maintains efficient convergence comparable to advanced initialization methods, while consistently outperforming standard LoRA. These results collectively highlight TLoRA as a highly efficient and effective fine-tuning strategy, delivering optimal performance across diverse tasks with reduced parameter overhead. Additionally, we demonstrate the effectiveness of TLORA across different model architectures and scales (e.g., LLaMA3 and Mistral) in Appendix D.1.

4.3 Ablation Study

We conduct an ablation study to assess the impact of each component. The experimental setup is detailed as follows:

- **LoRA:** The standard LoRA.
- **+ RA (Rank Adaptation):** The standard LoRA enhanced Rank Adaptation component.
- **+ SA (Scale factor Adaptation):** The standard LoRA enhanced Scale factor Adaptation component.
- **+ Init (Task-aware Initialization):** The baseline LoRA enhanced Task-aware Initialization component.
- **TLoRA:** Our full proposed model, which integrates Init, RA, and SA.

The results of the ablation study (Table 4) perfectly corroborate our theoretical framework regarding the tight coupling between "direction alignment" and "capacity allocation." First, we observe that under standard random initialization, applying only rank adaptation (+RA) or scale factor adaptation (+SA) yields minimal gains and can even degrade performance. This is precisely because random initialization fails to align with the task-relevant feature basis; blindly allocating rank capacity or scaling updates within this "wrong" subspace fails to effectively capture core task-relevant features. In stark contrast, introducing only our task-aware initialization (+Init), leading to a substantial performance leap across all benchmarks. The full TLORA model successfully maximizes the global optimization objective through adaptive rank and scaling allocation, ultimately achieving the best overall performance. Furthermore, as shown in Figure 2 (Right), this integrated approach accelerates model convergence, underscoring its superior efficiency.

Method	GSM8K	MATH	HumanEval	MBPP
LoRA	44.80	6.18	20.70	35.70
+ RA	45.33	5.96	22.60	34.40
+ SA	45.86	6.30	23.20	35.40
+ Init	51.78	7.74	22.00	39.40
+ Init + RA	54.05	7.68	22.60	38.60
+ Init + SA	55.11	8.36	22.00	39.40
+ RA + SA	47.68	6.50	22.60	36.50
TLoRA	**56.34**	**9.08**	**23.50**	**40.20**

Table 4: Our ablation studies evaluate performance across various experimental settings. The results presented are derived from fine-tuning the LLaMA2-7B model using the Code Feedback 100k and MetaMathQA 100k data subsets.

4.4 Expressiveness of the Frozen A Matrix

A core design choice in TLORA is freezing the projection matrix A post-initialization. Theoretically, freezing A confines the weight update strictly to its initial row space, thereby restricting dynamic "subspace evolution." A natural concern is whether this limitation caps the model's expressiveness upper bound, particularly in complex tasks (e.g., mathematical reasoning or dialogue alignment) where the optimal low-rank subspace might evolve during training.

To rigorously address this concern, we provide evidence from both static final performance and dynamic training trajectories. First, as shown in Table 5, the performance disparity is minimal across most benchmarks. Specifically, Frozen-A not only matched the performance of the unfrozen

Method	GSM8K	MATH	HumanEval	MBPP	MT-Bench
TLORA (Unfrozen-A)	**57.01**	8.78	23.20	**40.70**	5.09
TLORA (Frozen-A)	56.34	**9.08**	**23.50**	40.20	**5.13**

Table 5: Performance comparison between the standard TLORA (Frozen-A) and the fully trainable variant (Unfrozen-A) across math reasoning, code generation, and chat generation benchmarks. The results demonstrate that freezing A maintains competitive performance while halving the trainable parameters per adapter.

Task	Method	1000 Steps	2000 Steps	3086 Steps (Final)
GSM8K	TLORA (Frozen A)	59.05	62.47	64.36
	TLORA (Unfrozen A)	59.21	62.69	64.51
MATH	TLORA (Frozen A)	9.56	11.66	12.04
	TLORA (Unfrozen A)	9.26	11.92	13.08

Table 6: Performance of Frozen A vs. Unfrozen A on the MetaMathQA dataset across different training stages.

variant on most tasks but even surpassed it on the MATH (9.08% vs. 8.78%), MBPP (23.50% vs. 23.20%), and MT-bench (5.13 vs. 5.09) benchmarks.

Second, to directly validate the necessity of "subspace evolution," we conducted a detailed tracking experiment on the complete MetaMathQA dataset. We compared Frozen-A against the trainable Unfrozen-A variant, evaluating their performance every 1,000 steps (Table 6). The empirical data reveals that the performance gap between freezing and unfreezing A remains extremely marginal across all training stages. This compellingly demonstrates that the initial subspace captured by TLORA via W_0C is already near-optimal. Because this initialized subspace accurately aligns with the core feature directions required for complex reasoning from step zero, subsequent dynamic subspace evolution via gradient descent becomes largely redundant. Consequently, freezing A is not a restrictive compromise, but rather a deliberate and highly effective design choice that guarantees maximum parameter efficiency and training stability without sacrificing task performance.

4.5 Computational and Memory Analysis

To validate TLoRA's efficiency, we fine-tune Llama-2-7B on the MetaMathQA dataset, recording actual training time and memory consumption. TLoRA requires a one-time precomputing phase for initialization and importance allocation. To minimize memory peaks during this phase, we compute the covariance matrix layer by layer and immediately offload it to CPU memory. As reported in Table 7, the initialization process incurs merely 232 seconds, a negligible fraction of the approximately 5-hour total training duration. Furthermore, compared to LoRA, TLoRA freezes the projection matrix A, eliminating the need to store optimizer states for these parameters and significantly reducing GPU memory consumption during training.

Method	Stage	Time	Memory
LoRA	Initialization	-	-
	Training	4h48min15s	63530MB
TLoRA	Initialization	232.47s	17098 MB
	Training	4h49min23s	50448 MB

Table 7: Time and GPU memory costs during initialization and training.

5 Conclusion

This paper proposes TLoRA, a novel task-aware low-rank adaptation method that aligns adapters with task-relevant feature subspaces at the onset of training by performing SVD on the product of pre-trained weights and input activation covariance. Furthermore, to enhance parameter efficiency, we introduce a sensitivity-based importance scoring mechanism, enabling the adaptive allocation of ranks and scaling factors across different modules. Across extensive benchmarking, TLoRA consistently outperforms existing PEFT methods. Notably, since TLoRA operates solely during the initialization stage, it can be seamlessly integrated into existing LoRA pipelines as an efficient alternative initialization method, offering a simple yet effective enhancement to model adaptation.

Limitations

One limitation of this work stems from computational resource constraints, which restricted our evaluation primarily to the T5-Base and LLaMA2-7B models. Consequently, the scalability of TLoRA to larger-scale models (e.g., LLaMA2-70B) or Mixture-of-Experts (MoE) architectures remains to be empirically verified.

A second limitation lies in the scope of our application; while our method is theoretically applicable to other architectures such as Vision Transformers (ViTs) and Vision-Language Models (VLMs), this study focuses exclusively on Natural Language Processing (NLP) tasks.

Acknowledgements

This work was supported by the Guangdong Provincial Key Fields Special Project for Ordinary Universities (2025ZDZX1027).

References

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

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

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Luisa Bentivogli, Ido Dagan, Hoa T Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth pascal recognizing textual entailment challenge. In *Proceedings of the TAC*.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, and 1 others. 2024. Lora learns less and forgets less. *arXiv preprint arXiv:2405.09673*.

Kerim Büyükakyüz. 2024. Olora: Orthonormal low-rank adaptation of large language models. *arXiv preprint arXiv:2406.01775*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems, 2021. *URL https://arxiv.org/abs/2110.14168*, 9.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, and 1 others. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, and 1 others. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

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

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

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

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

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

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

Chen Liang, Simiao Zuo, Minshuo Chen, Haoming Jiang, Xiaodong Liu, Pengcheng He, Tuo Zhao, and Weizhu Chen. 2021. Super tickets in pre-trained language models: From model compression to improving generalization. *arXiv preprint arXiv:2105.12002*.

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

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

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

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

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

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2020. Adapterdrop: On the efficiency of adapters in transformers. *arXiv preprint arXiv:2010.11918*.

Reece Shuttleworth, Jacob Andreas, Antonio Torralba, and Pratyusha Sharma. 2024. Lora vs full fine-tuning: An illusion of equivalence. *arXiv preprint arXiv:2410.21228*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2023. Dylora: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3274–3287.

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

Hanqing Wang, Yixia Li, Shuo Wang, Guanhua Chen, and Yun Chen. 2025. Milora: Harnessing minor singular components for parameter-efficient llm finetuning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4823–4836.

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

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023a. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. 2023b. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*.

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

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

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

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

Qingru Zhang, Simiao Zuo, Chen Liang, Alexander Bukharin, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2022. Platon: Pruning large transformer models with upper confidence bound of weight importance. In *International conference on machine learning*, pages 26809–26823. PMLR.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

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

Jiacheng Zhu, Kristjan Greenewald, Kimia Nadjahi, Haitz Sáez De Ocáriz Borde, Rickard Brüel Gabrielsson, Leshem Choshen, Marzyeh Ghassemi, Mikhail Yurochkin, and Justin Solomon. 2024. Asymmetry in low-rank adapters of foundation models. *arXiv preprint arXiv:2402.16842*.

A Theoretical Derivation of Optimal Initialization

We theoretically analyze the effect of the selection of A on fine-tuning performance. Inspired by the literature (Zhu et al., 2024), we formalize the problem as follows: given a projection matrix A, when A is frozen and B is optimized under the least-squares criterion, the expected loss corresponding to the optimal solution B^* is denoted as $\mathcal{L}(A, B^*)$. Our objective is to identify the optimal A^* that minimizes this loss. This loss function can be expressed as:

$$\mathcal{L}(A, B^*) = d_{out}\sigma^2 + \mathrm{Tr}[\Delta C \Delta^\top] - \mathrm{Tr}[AC\Delta^\top \Delta C A^\top (ACA^\top)^{-1}] \tag{8}$$

Where $C = \mathrm{Cov}[X]$ denotes the covariance matrix of the input data, Δ represents the weight update under ideal conditions, and $d_{out}\sigma^2$ represents the irreducible error. Since the first two terms, $d_{out}\sigma^2$ and $\mathrm{Tr}[\Delta C \Delta^\top]$, depend solely on the data distribution and the target taskbeing independent of the selection of Aour objective of minimizing the loss $\mathcal{L}$ is equivalent to maximizing the trace of the third term:

$$J(A) = \mathrm{Tr}[AC\Delta^\top \Delta C A^\top (ACA^\top)^{-1}] \tag{9}$$

To ensure numerical stability and ensure non-singularity, we employ Tikhonov regularization by introducing a small damping factor ϵ (e.g., $\epsilon = 10^{-6}$): $C_{\text{reg}} = C + \epsilon I$. This modification guarantees that C_{reg} is strictly positive definite. For notational simplicity in the following steps, we will denote the regularized matrix C_{reg} simply as C.

Given that the covariance matrix C is symmetric positive definite, there exists a unique symmetric positive definite square root $C^{1/2}$ such that $C = C^{1/2}C^{1/2}$. Let us define the transformed matrix as $\tilde{A} = AC^{1/2}$. Leveraging the cyclic property of the trace ($\mathrm{Tr}(ABC) = \mathrm{Tr}(BCA)$), we can rewrite the objective function in terms of $\tilde{A}$:

$$J(\tilde{A}) = \mathrm{Tr}\left[\tilde{A}^\top (\tilde{A}\tilde{A}^\top)^{-1}\tilde{A} \cdot (\Delta C^{1/2})^\top (\Delta C^{1/2})\right] \tag{10}$$

Here, we decompose the optimization objective into the inner product of two matrices: $P_{\tilde{A}}$ and M_Δ.

- $P_{\tilde{A}} = \tilde{A}^\top(\tilde{A}\tilde{A}^\top)^{-1}\tilde{A}$ represents the orthogonal projection operator onto the row space of $\tilde{A}$, constrained to rank r..
- $M_\Delta = (\Delta C^{1/2})^\top(\Delta C^{1/2})$ is a symmetric positive semi-definite matrix determined by the data and the ideal update quantity, containing the target directional information for the task.

Consequently, the optimization problem is transformed into finding a projection matrix $P_{\tilde{A}}$ of rank r that maximizes $\mathrm{Tr}[P_{\tilde{A}} M_\Delta]$.

To solve this extremum problem, we introduce the von Neumann trace inequality. For any two symmetric matrices X and Y, the trace of their product satisfies the following upper bound:

$$|\mathrm{Tr}(XY)| \leq \sum_{i=1}^{d} \sigma_i(X)\sigma_i(Y) \tag{11}$$

Where $\sigma_i()$ denotes the eigenvalues of the matrix sorted in descending order. The necessary and sufficient condition for the equality to hold is that X and Y share the same eigenvectors. We apply this theorem to the current optimization objective, setting $X = P_{\tilde{A}}$ and $Y = M_\Delta$. Since $P_{\tilde{A}}$ is an orthogonal projection matrix of rank r, its eigenvalues $\sigma_i(P)$ have a specific distribution: $\sigma_i(P) = 1$ for $1 \leq i \leq r$, and 0 otherwise. Therefore, the inequality simplifies to:

$$\mathrm{Tr}(P_{\tilde{A}} M_\Delta) \leq \sum_{i=1}^{r} \sigma_i(M_\Delta) \tag{12}$$

This indicates that the maximum value of the objective function is determined by the sum of the top r largest eigenvalues of M_Δ. Based on the condition for equality in the inequality, the optimal projection matrix P^* must share eigenvectors with M_Δ. Specifically, P^* must be the projection onto the eigensubspace corresponding to the top r largest eigenvalues of M_Δ.

Let the eigendecomposition of M_Δ be $M_\Delta = V\Sigma V^\top$, where $V_r \in \mathbb{R}^{n\times r}$ denotes the matrix composed of the top r eigenvectors. Then the optimal projection matrix is uniquely determined as:

$$P^* = V_r V_r^\top \tag{13}$$

Recalling the definition $P = \tilde{A}^\top(\tilde{A}\tilde{A}^\top)^{-1}\tilde{A}$, we note that P depends solely on the row space of $\tilde{A}$ and not on the specific basis used to represent it. Consequently, any matrix $\tilde{A} = QV_r^T$ (where $Q \in \mathbb{R}^{r\times r}$ is invertible) will yield the same optimal projection matrix $P^* = V_r V_r^T$, resulting in the same optimal value for the objective function. Among all equivalent solutions, we select the canonical form where $Q = I_r$:

$$\tilde{A}^* = V_r^T \tag{14}$$

This derivation reveals a fundamental property: the theoretical optimal initialization is solely dependent on the principal directions (eigenvectors) of the target matrix M_Δ. Consequently, the row space of the optimal adapter $\tilde{A}^*$ is uniquely spanned by the top r right singular vectors of the target matrix M_Δ.

In practical fine-tuning scenarios, the true update quantity Δ is completely unknown before training, which makes the direct construction of the target matrix M_Δ impractical. However, the theoretical derivation above establishes a crucial principle: the optimal projection matrix P^* is solely determined by the principal eigendirections of the target matrix M_Δ, independent of the specific magnitude of its eigenvalues. This implies that we do not need to reproduce the specific numerical values of Δ; we only need to find a computable proxy matrix M_{proxy} that exhibits high alignment with M_Δ in terms of its "subspace directions."

To construct this proxy matrix, we draw upon a key insight from the LoRA (Hu et al., 2022): the weight update Δ resulting from fine-tuning exhibits a strong intrinsic correlation with the pre-trained weights W_0. Specifically, Δ does not randomly explore entirely new subspaces but instead tends to amplify certain directions that are already present in W_0 but were not fully emphasized. Building on this, we hypothesize that task-specific directions are essentially the pre-trained features (W_0) that are strongly activated by the current data (C). In other words, W_0 provides the potential feature basis, while C acts as the "Selector" for filtering these directions. Consequently, we define the computable proxy matrix as: $M_{\text{proxy}} = (W_0 C^{1/2})^\top(W_0 C^{1/2})$. This formula assumes that the principal subspace of M_{proxy} can serve as an accurate structural substitute for M_Δ.

To verify the feasibility of this hypothesis, we conducted an empirical analysis using the Llama2-7B model on the MetaMath dataset. Specifically, we measured the subspace overlap between the proxy matrix M_{proxy} and the ground-truth update

matrix M_Δ obtained via full fine-tuning. We adopted the subspace similarity metric proposed in LoRA to measure this alignment:

$$\phi(M_{\text{proxy}}, M_\Delta) = \frac{\|U_{\text{proxy}}^\top U_\Delta\|_F^2}{r} \in [0,1] \quad (15)$$

where U_{proxy} and U_Δ denote the top-r principal singular vectors of M_{proxy} and M_Δ, respectively. As illustrated in the figure, it depicts the subspace similarity between M_Δ and M_{proxy} computed from the Q projection matrix of each layer in Llama2-7B. There is a significant subspace consistency between M_{proxy} and M_Δ, with the average similarity reaching 0.71. This evidence strongly supports our hypothesis that M_{proxy} serves as an effective structural substitute for M_Δ. Finally, we

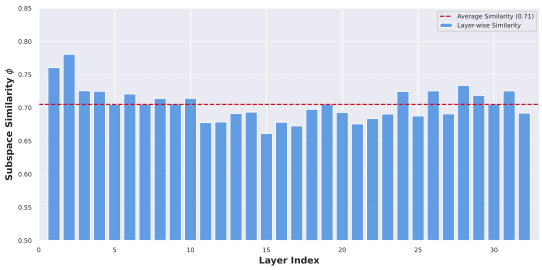

Figure 3: Subspace Similarity (ϕ).

map the optimization result back to the original parameter space via an inverse transformation to recover the LoRA projection matrix A. Recalling the transformation definition $\tilde{A} = AC^{1/2}$, by right-multiplying both sides of the equation by $C^{-1/2}$, we derive the theoretical optimal solution for A as:

$$A = V_r^T C^{-1/2} \in \mathbb{R}^{r \times n} \quad (16)$$

Where V_r is the matrix composed of the top r right singular vectors of the matrix $W_0 C^{1/2}$. This closed-form solution explicitly integrates both the pre-trained feature structure and data-driven activation statistics into the initialization of A.

B Empirical Analysis of Initialization Approximation

B.1 Numerical Instability of the Theoretical Solution

Under the constraint of a frozen matrix A, we derived the theoretically optimal initialization solution $A = V_r^T C^{-1/2}$, where V_r is obtained from the SVD of $WC^{1/2}$. However, in practice, TLoRA adopts a robust approximation strategy $A = V_r^T$, where V_r is derived from the SVD of WC. In this chapter, we will provide a detailed analysis of the instability inherent in the theoretical solution.

To investigate the root cause of the instability, we visualized the eigenvalues of the input activation covariance matrices for the q matrices in layers 5, 15, and 25 of Llama-2-7B. As illustrated in Figure 4, the spectra across various layers exhibit a consistent long-tail distribution. The vast majority of these are extremely small eigenvalues. Inverting these tail values would systemically amplify noise, justifying our use of the robust approximation $A = V_r^T$.

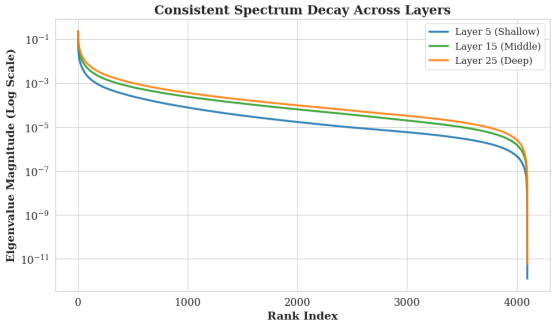

Figure 4: Eigenvalue spectrum consistency across varying layers.

To directly observe the instability of the theoretical solution, we compare the training performance of the theoretical solution with that of the approximate solution.

Figure 5 presents the gradient norm and training loss curves during the initial phase. The results provide compelling evidence: firstly, the theoretical solution exhibits a propensity for gradient explosion, with gradient norms exceeding those of TLoRA by several orders of magnitude. Secondly, this instability hinders effective model learning, resulting in a failure to converge. In contrast, TLoRA maintains stable and lower gradient norms, achieving rapid convergence.

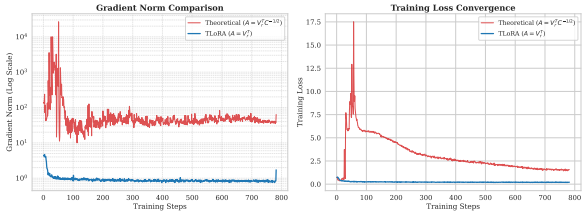

Figure 5: **(Left)** The gradient norm curves. **(Right)** The training loss curves.

B.2 Subspace Alignment Analysis

Although theoretical derivations suggest extracting a sub-space from $W_0C^{1/2}$, TLoRA actually performs SVD on W_0C. We adopt this approximation to circumvent the computational cost and potential numerical issues associated with root extraction, while leveraging covariance's superior noise suppression properties compared to its square root. To validate that replacing $W_0C^{1/2}$ with W_0C does not compromise the quality of the learned subspace, we conducted a quantitative analysis comparing structural alignment between the two objectives.

Specifically, we computed the subspace similarity ϕ between the top-r right singular vectors derived from W_0C (denoted as U_{Approx}) and those derived from $W_0C^{1/2}$ (denoted as U_{Theory}) across all layers of the LLaMA2-7B model. We utilized the projection metric defined as:

$$\phi(U_{Approx}, U_{Theory}) = \frac{1}{r}\|U_{Approx}^{\top} U_{Theory}\|_F^2 \tag{17}$$

As illustrated in Figure 6, the subspace similarity is consistently high across all layers, with an average similarity of 0.908. Even in the layers with the lowest overlap, the similarity score remains above 0.88. This strong alignment confirms that the principal directions extracted by W_0C are nearly identical to those of the theoretical solution.

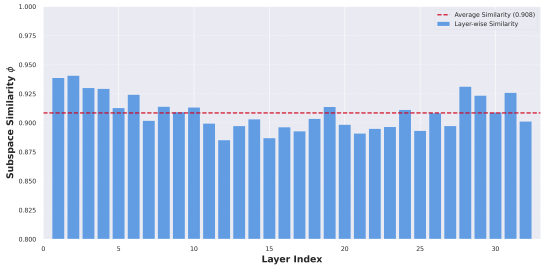

Figure 6: Subspace similarity analysis between the theoretical initialization target ($W_0C^{1/2}$) and our robust approximation (W_0C).

Furthermore, given the phenomenon observed in Section B.1 (Figure 4), using C (variance-weighted) rather than $C^{1/2}$ (standard deviation-weighted) proves more effective at suppressing the influence of noise directions and is therefore retained.

C Experiment Setups

To ensure reproducibility, we fixed the random seed to 42 for all experiments. Due to computational constraints, all reported results are derived from a single run.

C.1 GLUE benchmark

Hyperparameters	GLUE
Rank r	32
α	32
Dropout	0
Optimizer	AdamW
LR	3e-4
LR Scheduler	Cosine
Batch size	32
Warmup ratio	0.03
Epochs	3
Where	q, k, v, o, wi, wo
Sample size(TLoRA)	32

Table 8: Hyperparameter configurations of TLoRA and other PEFT methods for T5-Base on the GLUE Benchmark tasks.

C.2 Commonsense reasoning

Hyperparameters	Commonsense Reasoning
Rank r	32
α	32
Dropout	0
Optimizer	AdamW
LR	1e-4
LR Scheduler	linear
Batch size	32
Warmup ratio	0.03
Epochs	1
Where	q, k, v, up, down, o, gate
Sample size(TLoRA)	32

Table 9: Hyperparameter configurations of TLoRA and other PEFT methods for Llama2-7B on the Commonsense reasoning.

C.3 Math Code Chat

Hyperparameters	Math Code Chat
Rank r	128
α	128
Dropout	0
Optimizer	AdamW
LR	2e-5
LR Scheduler	cosine
Batch size	128
Warmup ratio	0.03
Epochs	1
Where	q, k, v, up, down, o, gate
Sample size(TLoRA)	32

Table 10: Hyperparameter configurations of TLoRA and other PEFT methods for Llama2-7B on the Math reasoning, Code generation, and chat generation.

Model	method	GSM8K	MATH
Mistral-7B	LoRA	70.81	19.40
	DoRA	70.58	18.68
	PISSA	73.16	19.64
	TLoRA	**73.38**	**20.32**
LLaMA2-13B	LoRA	56.10	9.84
	DoRA	57.16	9.78
	PISSA	62.92	12.24
	TLoRA	**64.51**	**12.74**
LLaMA3-8B	LoRA	71.94	9.84
	DoRA	72.10	22.22
	PISSA	76.26	24.06
	TLoRA	**77.86**	**24.76**

Table 11: Results of fine-tuning Mistral-7B, LlaMA2-13B, and LlaMA3-8B using TLoRA on a 100K subset of MetaMathQA. Bold numbers indicate the best performance achieved on this subtask.

D Analysis TLoRA

In this section, we conducted comprehensive analyses to validate the effectiveness of TLoRA. Unless otherwise specified, all analyses in this section are evaluated based on the fine-tuning results of the Llama-2-7B model on mathematical reasoning tasks.

D.1 Comparison with More Models

Models and Datasets. We further assessed TLoRA's robustness and scalability through experiments on three additional models: a larger model (LLaMA2-13B (Touvron et al., 2023)), a more advanced model(LLaMA3-8B), and a model from a different architecture (Mistral-7B (Jiang et al., 2023)). For these supplementary evaluations, we fine-tune the model on the same 100K subset of MetaMathQA (Yu et al., 2023) from our main experiments, focusing on the mathematical reasoning domain as an efficient yet challenging performance benchmark. Performance is evaluated on the GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021). Hyperparameters are kept consistent with the LLaMA2-7B setup. Detailed hyperparameter configurations for experiments are provided in Table 10.

Results. Table 11 presents the comparative results across Mistral-7B, LLaMA2-13B, and LLaMA3-8B. TLoRA consistently outperforms all baseline methodsLoRA, DoRA, and PISSAacross both GSM8K and MATH benchmarks for all three models. Notably, TLoRA demonstrates exceptional scalability on the larger LLaMA2-13B model, achieving 64.51% on GSM8K and 12.74% on MATH, surpassing the strong baseline PISSA by clear margins. Furthermore, on the more advanced LLaMA3-8B and architecturally distinct Mistral-7B, TLoRA maintains its leadership position, achieving the highest accuracy in every setting. These results strongly validate that TLoRA's effectiveness is robust across different model scales and architectural designs, consistently delivering superior fine-tuning performance compared to existing PEFT methods.

D.2 Comparison with Different initialization settings.

To validate the effectiveness of TLORA initialization, we conducted comparative ablation experiments across three distinct initialization settings. Specifically, while maintaining consistent subsequent training settings (i.e., freezing matrix A and training only matrix B), we compared the following three strategies for constructing feature extractor A:

- **Random-init.** Using the standard LoRA configuration with random initialization of A.
- **Weight-only, W-SVD.** Perform SVD decomposition on the pre-trained weights W_0 and initialize A using the first r principal singular vectors.
- **TLoRA-init, WC-SVD.** Follow the TLoRA-init initialization matrix A.

Analysis. As shown in Table 12, random-init performed the worst, which validates our theoretical motivation. Since the weight updates are strictly confined within the subspace spanned by the row vectors of A, a random initialization makes it hard to capture task-relevant feature directions, severely bottlenecking the model's adaptive capacity even if B is optimized. Second, while W-SVD improves performance by over 10% compared to the random baseline, it remains significantly inferior to the full TLoRA initialization. This gap suggests that weight magnitude in isolation is a suboptimal proxy for importance. Finally, WC-SVD (TLoRA) achieves the best results, outperforming W-SVD by a substantial margin of 8.34% on GSM8K and 1.90% on MATH. This empirical success confirms that the weighted product

W_0C more accurately captures the principal components of the task-specific update subspace, allowing the model to achieve superior adaptation performance with a frozen feature extractor.

Init	GSM8K	Math
Random-init	33.20	4.86
W-SVD	43.44	5.84
WC-SVD	51.78	7.74

Table 12: Results of fine-tuning llama2-7B using different init setting.

D.3 Sensitivity Analysis of Sample Size

Our task-aware initialization is a data-driven initialization method. A crucial issue is the sensitivity of our method to the number of samples used during initialization. To evaluate the robustness of TLoRA, we conduct a sensitivity analysis by varying the amount of data used for initialization and observing the impact on the performance of the final model.

Analysis. The results in the table 13 indicate that the performance of TLoRA is very robust to the number of samples used for calibration. In GSM8K and MATH benchmark tests, whether using small batches (such as 16 samples) or large batches (such as 512 samples), the final accuracy remains within a very narrow and high-performance range. This discovery strongly indicates that our task-aware initialization can effectively estimate high-quality adaptive subspaces from a small number of representative samples without requiring a large amount of calibration data. This result highlights the stability and practicality of TLoRA.

method	GSM8K	MATH
TLoRA(with 16 samples)	56.17	8.70
TLoRA(with 32 samples)	56.34	9.08
TLoRA(with 64 samples)	56.64	9.00
TLoRA(with 128 samples)	56.56	8.98
TLoRA(with 256 samples)	56.94	8.88
TLoRA(with 512 samples)	55.88	9.12

Table 13: Results of fine-tuning LlaMA2-7B using different sample size initialization TLoRA on a 100K subset of MetaMathQA.

D.4 Robustness of TLoRA towards different ranks

This section investigates the effect of varying rank configurations on the performance of TLoRA and LoRA. We evaluate the performance of the fine-tuned LLaMA2-7B model on math reasoning and code generation tasks. Table 14 illustrates the performance of both methods across different rank settings. Notably, TLoRA consistently outperforms LoRA in all configurations, demonstrating its effectiveness in improving fine-tuning performance and its robust generalization capabilities.

Method	Rank	GSM8K	MATH	MBPP	HumanEval
LoRA	16	32.90	4.22	32.80	15.90
	32	37.07	4.54	34.10	16.50
	64	40.48	5.42	36.50	17.70
	128	44.80	6.18	35.70	20.70
TLoRA	16	44.90	6.20	34.30	18.90
	32	47.80	6.64	37.30	19.50
	64	52.87	7.92	39.00	22.60
	128	56.35	9.08	39.20	23.50

Table 14: Comparison of LoRA and TLoRA with varying ranks for LLaMA2-7B on different tasks.

E Differentiated Module Importance Across Tasks

To further probe the task-specific nature of module importance, we analyse the difference in scores between the mathematical reasoning and code generation domains. We subtract the importance score of each module on the code task from its importance score on the math task. The resulting importance difference, visualised in a heatmap in Figure 7, reveals a highly localised pattern rather than a global shift. We observe a distinct difference in the middle layers of the network (approximately layers 8-15), where the v_{proj} and o_{proj} modules exhibit significantly higher importance for math reasoning than for code generation. In contrast, the vast majority of other modules across all layers show negligible differences, suggesting their roles are largely task-agnostic.

This findingthat only a sparse, specific subset of modules is highly task-sensitivepowerfully underscores the necessity of an adaptive allocation strategy like TLoRA. It demonstrates that superior performance hinges on the ability to identify and concentrate the parameter budget onto these few critical, task-specific modules, rather than uniformly distributing it across the many general-purpose

ones.

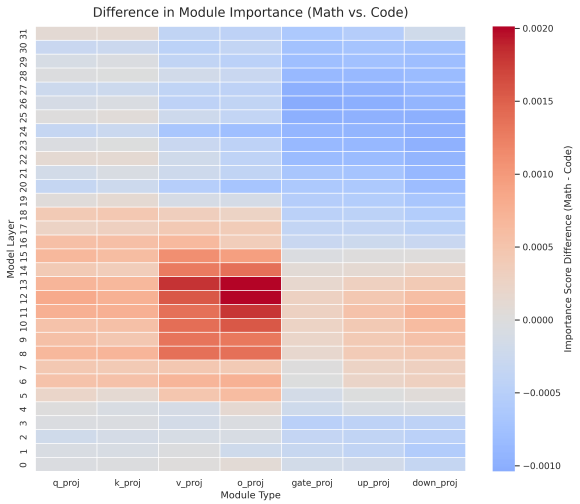

Figure 7: The data on the graph shows the difference in importance scores between mathematical tasks and code tasks.

F Baseline

Baselines. We compare TLoRA with several baselines in Experiments:

- **FULL** Fine-tuning the model with all parameters.
- **LoRA**(Hu et al., 2022) approximates weight ΔW updates through the product of two trainable low-rank matrices A and B.
- **AdaLoRA**(Zhang et al., 2023b) enhances performance by dynamically allocating the parameter budget across layers based on their importance.
- **LoRA+**(Hayou et al., 2024) improves upon LoRA by setting a higher learning rate for the adapter matrix B than for matrix A.
- **DoRA**(Liu et al., 2024) decompose the pre-trained weight matrix W into magnitude and direction component.
- **PiSSA**(Meng et al., 2024) initializes adapter matrices A and B using the principal singular components of the original weight matrix W.
- **OLoRA**(Büyükakyüz, 2024) initializes adapter matrices A and B using the orthogonal bases of the original weight matrix W.
- **LoRA-GA**(Wang et al., 2024) aligns the low-rank adaptation with the gradient direction of the pre-trained weights by using a gradient-based approximation to initialize matrix A and B.
- **CorDA**(Yang et al., 2024) constructs a task-relevant covariance matrix to identify important directions in the weight space, initializing the adapter based on the singular vectors of these covariance-weighted features.