TaRA: Training-Aware Low-Rank Adaptation Initialization

Anonymous ACL submission

Abstract

Low-Rank Adaptation (LoRA) has become a de facto standard for parameter-efficient fine-tuning (PEFT), yet its performance is highly sensitive to initialization due to the information bottleneck imposed by low-rank decomposition. Existing approaches attempt to construct high-quality LoRA initializations by exploiting principal components of pretrained weights, activations, or gradients. However, these methods do not directly account for the training dynamics of the full-rank model, which can lead to suboptimal convergence. In this paper, we propose Training-aware Low-Rank Adaptation Initialization (TaRA), a method that initializes LoRA such that the gradients induced by the low-rank factors closely approximate the gradient of the corresponding full-rank weight matrix. Derived from a principled mathematical formulation, TaRA improves gradient fidelity at the start of training while introducing negligible computational overhead. Across diverse and challenging fine-tuning tasks, TaRA consistently outperforms prior state-of-the-art methods, establishing a simple, robust, and scalable solution for effective LoRA initialization.

1 Introduction

Parameter-efficient fine-tuning (PEFT) has significantly reduced the cost of adapting large language models (LLMs), enabling their widespread deployment in practical applications (Houlsby et al., 2019; Li and Liang, 2021). Among PEFT methods, Low-Rank Adaptation (LoRA) (Hu et al., 2022) has emerged as the most widely adopted approach due to its simplicity, substantial reduction in fine-tuning resource requirements, and absence of inference-time overhead, as the learned updates can be merged into the base weights after training. These advantages have established LoRA as a de facto standard for efficient LLM fine-tuning.

Despite these benefits, LoRA introduces an inherent trade-off: the structured bottleneck imposed by low-rank decomposition alters the optimization trajectory relative to full-rank training, which can lead to suboptimal convergence even when model capacity is sufficient. Consequently, improving the effectiveness of LoRA without sacrificing its efficiency has become an important research direction (Meng et al., 2024; Buyukakyuz, 2024; Hayou et al., 2024).

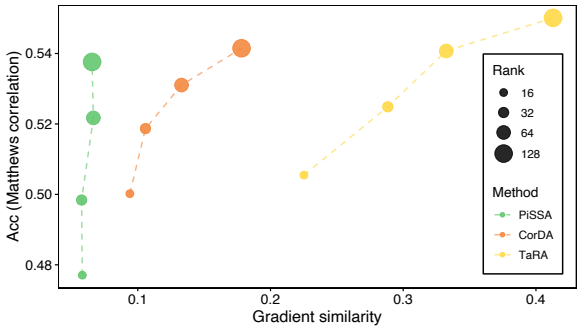


Figure 1: Gradient similarity, measured as the cosine similarity between the one-step gradient of the combined LoRA adapter and that of the full weight, correlates with downstream accuracy.

One promising direction is to design more informed initialization strategies for the low-rank adapters. While the original LoRA initializes its adapters with random or zero values, subsequent studies have shown that low-rank initializations guided by pretrained weight distributions (e.g., PiSSA (Meng et al., 2024)), joint weight–data statistics (e.g., CorDA (Yang et al., 2024)), or gradients from a single forward–backward pass (e.g., LoRA-GA, LoRA-One (Wang et al., 2024; Zhang et al., 2025)) consistently yield improved fine-tuning performance. These methods preserve the practical advantages of LoRA while alleviating degradation caused by low-rank constraints.

To further address this limitation, we propose Training-aware Low-Rank Adaptation (**TaRA**), a new LoRA initialization method. Unlike prior approaches that approximate tensor statistics, TaRA is

designed to preserve the update direction of full-weight training, allowing PEFT to better reproduce the learning dynamics of full fine-tuning. Grounded in a principled formulation, TaRA derives a low-rank initialization that best approximates full-weight updates by jointly leveraging activation covariance, gradient covariance, and pretrained weights from a single forward–backward pass. As illustrated in Figure 1, this initialization achieves substantially higher gradient alignment than prior methods at the same rank, leading to consistent downstream improvements. Extensive experiments on challenging reasoning tasks, including mathematical problem solving, code generation, and commonsense reasoning, demonstrate that TaRA outperforms existing approaches.

2 Prior Work on LoRA Initialization

In the original LoRA, the input-side factor matrix $\mathbf{A}$ is randomly initialized, while the output-side factor matrix $\mathbf{B}$ is set to zero. Although this design ensures that fine-tuning starts from the pretrained representations, the resulting initialization contains no meaningful task-related information, which can lead to slow early-stage optimization and suboptimal convergence (Wang et al., 2024).

Motivated by these limitations, numerous studies have explored improved initialization strategies for LoRA (Figure 2). Although the decomposed path produces a non-zero update, fine-tuning can still start from the pretrained weight $\mathbf{W}_0$ by absorbing this modification into a frozen residual weight, defined as $\mathbf{W}_{res} = \mathbf{W}_0 - \mathbf{AB}$. With an appropriate choice of $\mathbf{A}$ and $\mathbf{B}$, a properly initialized model can accelerate training and typically converges to better solutions than random initialization.

A key early work in this line of research is PiSSA (Meng et al., 2024), which proposes a data-agnostic initialization that leverages the pretrained weight matrix $\mathbf{W}_0 \in \mathbb{R}^{dout \times d_{in}}$. Specifically, PiSSA performs singular value decomposition (SVD), $\mathbf{W}_0 = \mathbf{USV}^\top$, and initializes LoRA using the top-r singular components:

$$\mathbf{A} = \mathbf{U}[:, :r]\mathbf{S}^{1/2}[:r, :r] \in \mathbb{R}^{dout \times r}, \quad (1)$$

$$\mathbf{B} = \mathbf{S}^{1/2}[:r, :r]\mathbf{V}^\top[:, :r] \in \mathbb{R}^{r \times d_{in}}. \quad (2)$$

More recently, CorDA (Yang et al., 2024) extends this idea by incorporating input statistics in addition to pretrained weights. CorDA first estimates the input activation covariance $\mathbf{\Sigma}_X = \mathbf{XX}^\top$ from input activations $\mathbf{X}$, and then performs the following decomposition:

$$\mathrm{SVD}(\mathbf{W}_0\mathbf{\Sigma}_X)\mathbf{\Sigma}_X^{-1} = \bar{\mathbf{U}}\bar{\mathbf{S}}\bar{\mathbf{V}}^\top. \quad (3)$$

The resulting components are used to initialize $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{W}_{res}$ in the same manner as PiSSA. Both PiSSA and CorDA align the LoRA subspace with principal components derived from pretrained weights or input-dependent statistics, resulting in faster convergence and improved accuracy.

Orthogonal to these approaches, LoRA-GA (Wang et al., 2024) and its successor, LoRA-One (Zhang et al., 2025), leverage gradient information. These methods are motivated by the observation that the fine-tuning quality of LoRA depends on the alignment of its update direction with the dominant singular subspace of the full fine-tuning gradient. Based on this insight, LoRA-One performs SVD of the one-step full gradient of weight $\mathbf{G}$,

$$\mathrm{SVD}(-\mathbf{G}) = \hat{\mathbf{U}}\hat{\mathbf{S}}\hat{\mathbf{V}}^\top, \quad (4)$$

and use the top-r singular components to initialize the LoRA. By leveraging gradient information, these methods initialize the LoRA modules along directions that are partially informed by training signals, often leading to improved fine-tuning performance in practice.

However, despite utilizing internal tensor statistics of the pretrained model, prior approaches primarily focus on enhancing the representational capacity of the low-rank decomposition, rather than explicitly aligning it with the learning dynamics of full fine-tuning. Consequently, a substantial performance gap remains, which cannot be effectively closed by merely increasing the rank or tuning hyperparameters.

3 Proposed Idea: TaRA

In this section, we introduce TaRA, a novel LoRA initialization method designed to enhance fine-tuning quality under a strict low-rank constraint. We begin by formalizing the objective, derive a principled low-rank solution, and then describe its practical implementation.

3.1 Motivation

The key motivation behind TaRA is simple yet fundamental: even under the low-rank bottleneck imposed by LoRA, we aim to faithfully approximate the training behavior of full fine-tuning. This perspective naturally leads to the following question:

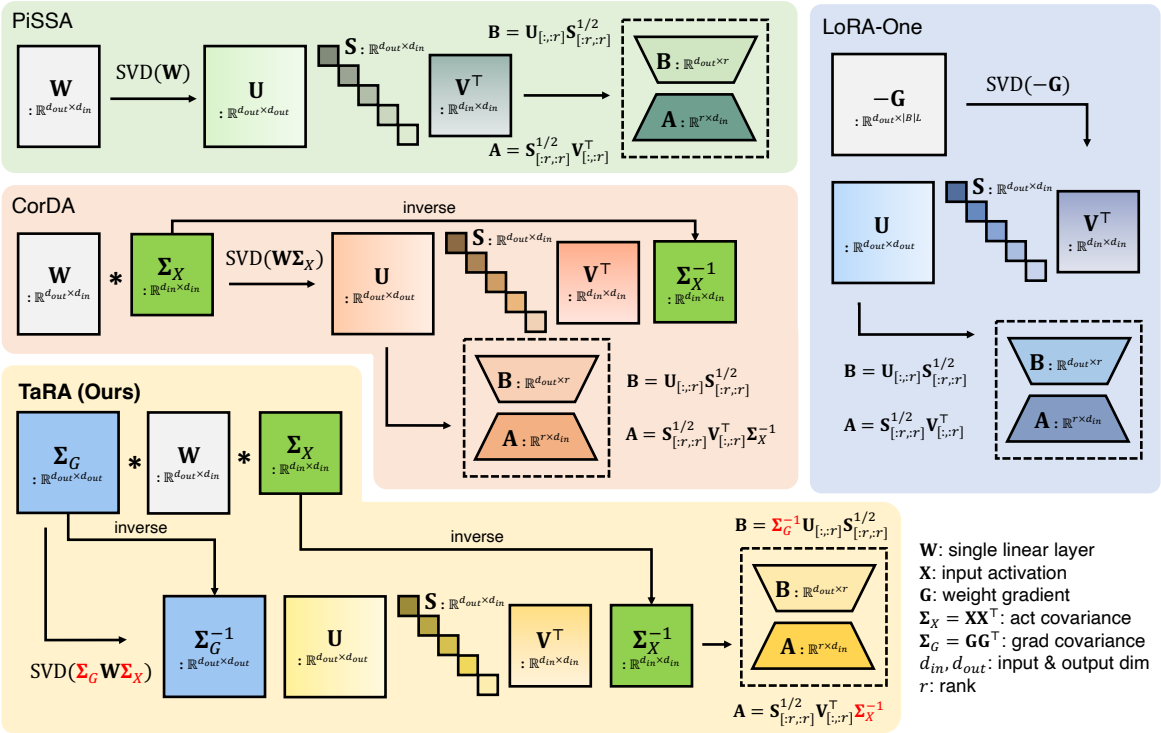


Figure 2: Illustration of our single-layer procedure. We collect activations $\mathbf{X}$ and gradients $\mathbf{G}$ to form covariances $\mathbf{\Sigma}_X$ and $\mathbf{\Sigma}_G$. PiSSA applies SVD to $\mathbf{W}$; CorDA applies SVD to $\mathbf{W\Sigma}_X$ and maps back with $\mathbf{\Sigma}_X^{-1}$; LoRA-One applies SVD to $-\mathbf{G}$. In contrast, TaRA applies SVD to $\mathbf{\Sigma}_G\mathbf{W\Sigma}_X$ and maps back with $\mathbf{\Sigma}_G^{-1}$ and $\mathbf{\Sigma}_X^{-1}$.

Under a rank-r constraint, how can a low-rank update best approximate the full fine-tuning update direction?

To formalize this intuition, we consider the following objective:

$$\tilde{\theta} = \arg\min_{\theta} \|\nabla\mathcal{L}(\theta) - \nabla\mathcal{L}(\theta_0)\|_F^2, \quad \text{s.t. } \operatorname{rank}(\theta) \le r, \tag{5}$$

where $\mathcal{L}$ denotes the task loss. Intuitively, this objective seeks a rank-constrained update that best approximates the full fine-tuning gradient at θ_0, thereby aligning the update direction.

3.2 Training-Relevant Decomposition

To obtain a tractable formulation, we adopt a standard second-order view of the loss landscape around the initialization point. In particular, we focus on the one-step gradient at initialization (Wang et al., 2024), which captures the dominant training signal in the early stage of optimization. In practice, this approximation is sufficient to guide effective low-rank initialization, as demonstrated by the convergence results in Section 4.2.

Using a second-order Taylor expansion of the loss around θ_0, we obtain

$$\mathcal{L}(\theta) \approx \mathcal{L}(\theta_0) + \nabla\mathcal{L}(\theta_0)^\top(\theta - \theta_0) + \frac{1}{2}(\theta - \theta_0)^\top \mathbf{H}(\theta - \theta_0), \tag{6}$$

where $\mathbf{H}$ denotes the Hessian evaluated at θ_0.

In large models, directly computing the Hessian is infeasible. Following common practice, we instead use the Fisher information matrix $\mathcal{F}$ as a tractable surrogate for local curvature (Chekalina et al., 2025; Martens, 2020). Using the K-FAC factorization (Martens and Grosse, 2015), the Fisher matrix can be written as

$$\mathcal{F} \approx \mathbf{\Sigma}_X \otimes \mathbf{\Sigma}_G, \tag{7}$$

where $\mathbf{\Sigma}_G = \mathbf{GG}^\top$ denotes the gradient covariance. Under this formulation, differentiating Eq. (6) with respect to θ yields (derivation is in Appendix):

$$\nabla\mathcal{L}(\theta) - \nabla\mathcal{L}(\theta_0) \approx \mathbf{\Sigma}_G(\theta - \theta_0)\mathbf{\Sigma}_X. \tag{8}$$

This relation reveals a key insight: the change in gradient induced by a parameter update is modulated by the curvature structure of the loss. Consequently, directions associated with large Fisher or Hessian values have a disproportionately large

influence on the optimization trajectory. Preserving these directions is therefore crucial for faithfully approximating full fine-tuning under a low-rank space.

Empirically, the Hessian spectrum of modern neural networks is known to exhibit a highly structured form, consisting of a small set of prominent outlier eigenvectors and a near-zero bulk that is largely orthogonal to them (Gur-Ari et al., 2018; Sagun et al., 2017). Moreover, prior work shows that gradient descent effectively evolves within a small subspace spanned by the top Hessian eigenvectors (Gur-Ari et al., 2018). These observations suggest that dominant curvature directions provide a natural criterion for identifying effective low-rank training updates.

Motivated by this insight, we seek a rank-r approximation $\tilde{\theta}$ that minimizes the curvature-weighted gradient deviation in Eq. (8):

$$\tilde{\theta} \approx \arg\min_{\theta} \|\mathbf{\Sigma}_G(\theta - \theta_0)\mathbf{\Sigma}_X\|_F^2, \quad \text{s.t. } \operatorname{rank}(\theta) \leq r. \tag{9}$$

This corresponds to a classical low-rank matrix approximation problem. By the Eckart–Young theorem (Eckart and Young, 1936; Golub and Van Loan, 2013), the optimal solution is obtained by retaining the top-r singular components of decomposition:

$$\tilde{\theta} \approx \mathbf{\Sigma}_G^{-1} \operatorname{SVD}_r(\mathbf{\Sigma}_G \theta_0 \mathbf{\Sigma}_X) \mathbf{\Sigma}_X^{-1}, \tag{10}$$

where $\operatorname{SVD}_r(\cdot)$ denotes the truncated SVD.[1]

This result shows that the optimal low-rank update corresponds to preserving the dominant curvature directions of the pretrained parameter θ_0.

3.3 Implementation of TaRA

Based on the above formulation, TaRA can be implemented as an efficient layer-wise LoRA initialization procedure for linear layers, summarized in Algorithm 1. In the **Collect** stage, we gather task-dependent statistics using a small calibration dataset D. During the forward pass, we record input activations and concatenate them to form $\mathbf{X} \in \mathbb{R}^{d_{\text{in}} \times |B|L}$. During the backward pass, we collect the corresponding weight gradients to obtain $\mathbf{G} \in \mathbb{R}^{d_{\text{out}} \times |B|L}$. These statistics are used to estimate the covariance matrices $\mathbf{\Sigma}_X$ and $\mathbf{\Sigma}_G$. In the

[1]This is often confused with Fisher-weighted model compression; key differences are described in Appendix E.

Algorithm 1 TaRA, init for single layer

Input: Pre-trained weight of single linear layer $\mathbf{W}_0 \in \mathbb{R}^{d_{out} \times d_{in}}$, calibration dataset D, LoRA rank $r \in \mathbb{N}$

Collect:

1: Collect input activations in forward pass $\mathbf{X} \in \mathbb{R}^{d_{in} \times |B|L}$

2: Collect weight gradients in backward pass $\mathbf{G} \in \mathbb{R}^{d_{out} \times |B|L}$

3: Compute activation covariance $\mathbf{\Sigma}_X \leftarrow \mathbf{X}\mathbf{X}^\top \in \mathbb{R}^{d_{in} \times d_{in}}$

4: Compute gradient covariance $\mathbf{\Sigma}_G \leftarrow \mathbf{G}\mathbf{G}^\top \in \mathbb{R}^{d_{out} \times d_{out}}$

Init:

1: $\tilde{\mathbf{U}}, \tilde{\mathbf{S}}, \tilde{\mathbf{V}}^\top \leftarrow \operatorname{SVD}(\mathbf{\Sigma}_G \mathbf{W}_0 \mathbf{\Sigma}_X)$

2: $\mathbf{A} \leftarrow \mathbf{\Sigma}_G^{-1} \tilde{\mathbf{U}}_{[:,:r]} \tilde{\mathbf{S}}_{[:r,:r]}^{1/2} \in \mathbb{R}^{d_{out} \times r}$

3: $\mathbf{B} \leftarrow \tilde{\mathbf{S}}_{[:r,:r]}^{1/2} \tilde{\mathbf{V}}_{[:,:r]}^\top \mathbf{\Sigma}_X^{-1} \in \mathbb{R}^{r \times d_{in}}$

4: $\mathbf{W}_{res} \leftarrow \mathbf{W}_0 - \mathbf{A}\mathbf{B} \in \mathbb{R}^{d_{out} \times d_{in}}$

Init stage, we compute the SVD of $\mathbf{\Sigma}_G \mathbf{W}_0 \mathbf{\Sigma}_X$ and project the resulting components back to the original space using $\mathbf{\Sigma}_G^{-1}$ and $\mathbf{\Sigma}_X^{-1}$. This produces singular components that capture the training-relevant directions under the Fisher-weighted metric.

In practice, the covariance matrices may be rank-deficient (Yankun et al., 2025; Zhao et al., 2024), which makes direct inversion unstable. To ensure numerical stability, we apply diagonal damping, $\mathbf{\Sigma}_G \leftarrow \mathbf{\Sigma}_G + c\beta\mathbf{I}$ and $\mathbf{\Sigma}_X \leftarrow \mathbf{\Sigma}_X + c\beta\mathbf{I}$, a well-known technique in second-order and Fisher-based optimization methods (Martens and Grosse, 2015; Ledoit and Wolf, 2012). Following (Yang et al., 2024), we set β to the mean singular value of each covariance matrix and use $c = 10^{-2}$ (see Appendix C for further analysis).

After computing the decomposition, we truncate it to rank r and construct the LoRA factors. The trainable matrices $\mathbf{A}$ and $\mathbf{B}$ are initialized from the top-r singular components, scaled by the inverse covariance matrices to align them with the curvature-weighted subspace. We then update the frozen residual weight so that the model initially preserves the pretrained function. Applying this procedure independently to each linear layer yields a training-friendly LoRA initialization.

4 Experiments

We evaluate the proposed method through extensive experiments across multiple LLMs and a diverse set of downstream tasks. We compare TaRA against full fine-tuning and several PEFT

Rank	Method	GSM8k	GSM8k-COT	MATH	HumanEval	MBPP	AVG
	Full fine-tuning (lr=4e-5)	54.71 ± 0.20	50.70 ± 0.25	10.15 ± 0.04	26.83 ± 0.50	26.20 ± 0.65	33.72 ± 0.33
128	LoRA (lr=4e-5)	44.83 ± 0.38	37.23 ± 0.55	6.12 ± 0.04	21.14 ± 0.29	22.67 ± 0.77	26.40 ± 0.41
	PiSSA (lr=4e-5)	53.85 ± 0.07	47.42 ± 0.14	8.98 ± 0.04	23.46 ± 0.64	24.87 ± 0.57	31.72 ± 0.29
	CorDA (lr=4e-5)	55.54 ± 0.54	47.56 ± 0.27	9.41 ± 0.11	**24.06** ± 0.20	24.10 ± 0.22	32.13 ± 0.27
	LoRA-One (lr=2e-4)	52.92 ± 0.49	46.74 ± 0.22	9.28 ± 0.07	23.81 ± 0.59	24.83 ± 0.47	31.52 ± 0.37
	TaRA (lr=4e-5)	**56.59** ± 0.25	**50.42** ± 0.13	**10.08** ± 0.10	22.59 ± 0.58	**25.20** ± 0.67	**32.98** ± 0.34
64	LoRA (lr=4e-5)	40.31 ± 0.18	32.65 ± 0.20	5.27 ± 0.05	17.88 ± 0.60	21.40 ± 0.59	23.50 ± 0.32
	PiSSA (lr=4e-5)	49.05 ± 0.86	43.57 ± 0.28	7.29 ± 0.16	21.47 ± 0.97	23.73 ± 0.62	29.02 ± 0.58
	CorDA (lr=4e-5)	51.45 ± 0.40	42.77 ± 0.33	8.13 ± 0.18	21.14 ± 0.76	24.93 ± 0.41	29.68 ± 0.42
	LoRA-One (lr=2e-4)	51.54 ± 0.83	42.84 ± 0.43	8.06 ± 0.08	**24.39** ± 0.86	25.20 ± 0.43	30.41 ± 0.53
	TaRA (lr=4e-5)	**54.12** ± 0.93	**44.49** ± 0.11	**9.23** ± 0.07	23.58 ± 0.70	**25.40** ± 0.36	**31.36** ± 0.43
32	LoRA (lr=4e-5)	38.36 ± 0.35	31.51 ± 0.22	5.10 ± 0.12	18.90 ± 0.68	20.47 ± 0.82	22.87 ± 0.44
	PiSSA (lr=4e-5)	46.40 ± 0.06	36.64 ± 0.29	6.42 ± 0.06	20.73 ± 0.50	22.31 ± 0.13	26.50 ± 0.21
	CorDA (lr=4e-5)	48.09 ± 0.22	38.82 ± 0.64	6.96 ± 0.06	19.10 ± 0.58	24.33 ± 0.82	27.46 ± 0.46
	LoRA-One (lr=2e-4)	47.44 ± 0.30	41.55 ± 0.17	7.49 ± 0.11	20.69 ± 0.26	24.60 ± 0.43	28.35 ± 0.26
	TaRA (lr=4e-5)	**50.14** ± 0.08	**41.70** ± 0.27	**8.57** ± 0.06	**21.53** ± 0.29	**24.73** ± 0.52	**29.33** ± 0.24

Table 1: Comparison of LoRA, PiSSA, CorDA, LoRA-One, and TaRA on natural language generation tasks. Each result is the mean over three seeds, and the standard deviation is shown in small gray text. Red color indicates the best PEFT result within each model block, excluding Full FT.

baselines, including LoRA (Hu et al., 2022), PiSSA (Meng et al., 2024), CorDA (Yang et al., 2024), and LoRA-One (Zhang et al., 2025). Our evaluation covers two task categories: natural language generation (NLG) and natural language understanding (NLU), and systematically examines performance across different model and LoRA rank budgets.

4.1 Experimental Setup

Natural Language Generation Tasks. For NLG, we consider math problem solving and code generation. We fine-tune LLaMA-2-7B (Touvron et al., 2023) on 100K samples from MetaMathQA (Yu et al., 2023) for math and 100K samples from CodeFeedback-Filtered-Instruction (Zheng et al., 2024) for code. Models are trained with LoRA ranks of 128, 64 and 32 to assess robustness across rank budgets (LoRA α is set equal to the rank).

For evaluation, math performance is measured on GSM8K-D (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) using direct prompting, as well as GSM8K-COT (Wei et al., 2022) under an 8-shot chain-of-thought setting with the Language Model Evaluation Harness (Biderman et al., 2024). Code generation is evaluated on HumanEval (Chen, 2021) and MBPP (Austin et al., 2021) using the BigCode Evaluation Harness (Ben Allal et al., 2022). All evaluation harnesses are used with their default configurations.

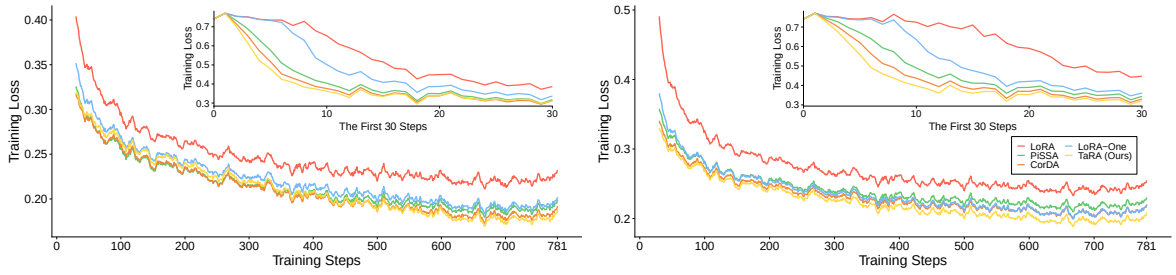
Natural Language Understanding Tasks. For NLU, we focus on commonsense reasoning. Models are fine-tuned on the Commonsense-170K dataset (Hu et al., 2023). To evaluate scalability and generality, we conduct experiments on three base models—DeepSeek-R1-Distill-Qwen-1.5B (Guo et al., 2025), LLaMA-2-7b (Touvron et al., 2023), LLaMA-3.1-8B (Grattafiori et al., 2024), and Qwen-3-8B (Yang et al., 2025)—with a fixed LoRA rank of 128 (LoRA α is set equal to the rank).

We evaluate performance on eight benchmarks: BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-Challenge, ARC-Easy (Clark et al., 2018), and OBQA (Mihaylov et al., 2018).

For full fine-tuning, LoRA, PiSSA, and CorDA, we follow the hyperparameter settings of CorDA (Yang et al., 2024) and select the best learning rate. For LoRA-One, which is sensitive to learning-rate choice, we use the optimal value reported in its original paper. For all methods that require calibration (CorDA, LoRA-One, and TaRA), we set the calibration set size to 256 and conducted all experiments under this setting. All results are obtained from the final training checkpoint and reported in terms of task accuracy. Additional details are provided in the Appendix. All experiments are conducted on A100 GPUs (80GB).

4.2 Natural Language Generation Task

Table 1 compares performance on natural language generation benchmarks across different LoRA ranks ($r \in 128, 64, 32$). Overall, TaRA achieves the highest average performance at all ranks, consistently outperforming prior initialization methods such as PiSSA, CorDA, and LoRA-One. At



(a) $r = 128$ (b) $r = 32$

Figure 3: Training loss over steps on LLaMA-2-7B fine-tuned with MetaMathQA. We compare LoRA, PiSSA, CorDA, LoRA-One, and TaRA at (a) $r = 128$ and (b) $r = 32$; insets show the first 30 steps.

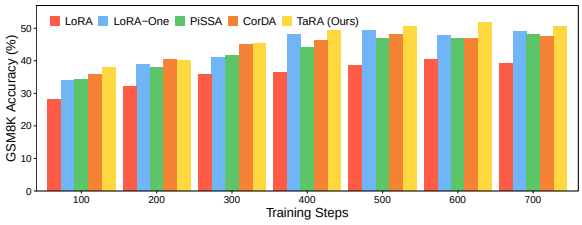


Figure 4: GSM8K-D accuracy over training steps when fine-tuning LLaMA-2-7B on the MetaMathQA dataset.

$r = 128$, TaRA attains the best results on GSM8K-D/GSM8K-COT, MATH, and MBPP, and also yields the highest average score. At $r = 64$, TaRA remains the top-performing method on GSM8K-D/GSM8K-COT, MATH, and MBPP, leading to the best average performance as well. Even at $r = 32$, TaRA exhibits a relatively small performance drop compared to other baselines, achieving the best performance across all tasks and demonstrating the most stable behavior across ranks. These results highlight the strength of the proposed initialization in efficiently capturing training-relevant directions under a constrained parameter budget.

Notably, LoRA-One also underperforms PiSSA and CorDA at $r = 128$, suggesting that directly applying SVD to gradients is ineffective at higher ranks. Overall, these results indicate that neither simple SVD-based nor random initialization is sufficient to capture training-relevant directions under tight constraints, whereas TaRA does so effectively.

Figures 3a and 3b show the training loss curves on MetaMathQA for $r = 128$ and $r = 32$. In both settings, TaRA converges more smoothly and reaches a lower loss than prior methods. This suggests that the proposed initialization provides a more effective update subspace at the early stage of training, enabling more efficient optimization, and that its benefit persists into later phases of training (Gur-Ari et al., 2018). This advantage persists throughout fine-tuning, as also reflected in GSM8K-D accuracy over training steps (Figure 4). While PiSSA and CorDA exhibit limited improvement and LoRA-One improves only in later stages, TaRA achieves rapid early gains and consistently attains the best performance at both intermediate and final stages.

4.3 Natural Language Understanding Task

Table 2 compares performance on standard commonsense reasoning benchmarks across three models: DeepSeek-R1-Distill-Qwen-1.5B, LLaMA-2-7B LLaMA-3.1-8B, and Qwen-3-8B. Overall, TaRA achieves the best average performance on DeepSeek-R1-Distill-Qwen-1.5B, LLaMA-2-7B and LLaMA-3.1-8B, outperforming PiSSA, CorDA, and LoRA-One. On Qwen-3-8B, TaRA slightly underperforms LoRA-One in terms of average score but remains competitive, consistently outperforming LoRA, PiSSA, and CorDA. While fine-tuning performance is often distribution-dependent and no method consistently dominates across all settings, a well-known phenomenon also observed in prior work, we evaluate methods based on their generality across diverse scenarios. Under this criterion, TaRA consistently performs well, achieving top-1 performance in 15 out of 32 cases and second-best in 11 more. In contrast, LoRA-One and CorDA achieve top-1 in only 8 and 5 cases, respectively, highlighting the robustness of TaRA beyond specific models or tasks.

4.4 Gradient Alignment Analysis

To verify whether the proposed method can effectively extract principal components that are *important from a training perspective*, we use the one-step gradient computed from the full-rank weight $\mathbf{W}_0$ as a reference. We apply PiSSA, CorDA, and TaRA to obtain rank-r approximations, restore

Model	Method	BoolQ	PIQA	SIQA	HS	WG	ARC-c	ARC-e	OBQA	AVG
DeepSeek-R1 -Distill-Qwen -1.5B	Full FT	63.04 ± 0.12	61.41 ± 0.38	52.58 ± 0.02	27.40 ± 0.10	49.41 ± 0.49	45.22 ± 0.36	61.38 ± 0.16	42.47 ± 0.62	50.36 ± 0.18
	LoRA	62.60 ± 0.04	61.64 ± 0.38	51.66 ± 0.16	25.77 ± 0.24	50.57 ± 0.44	46.33 ± 0.30	62.16 ± 0.24	42.73 ± 0.52	50.43 ± 0.05
	PiSSA	64.68 ± 0.11	65.02 ± 0.09	63.15 ± 0.27	**38.84** ± 0.31	57.75 ± 0.16	54.38 ± 0.42	69.32 ± 0.03	54.60 ± 0.16	58.47 ± 0.03
	CorDA	**64.89** ± 0.13	66.14 ± 0.11	65.51 ± 0.11	37.04 ± 0.28	58.27 ± 0.23	**55.26** ± 0.15	68.62 ± 0.19	56.20 ± 0.49	58.99 ± 0.09
	LoRA-One	63.77 ± 0.18	64.33 ± 0.27	65.01 ± 0.34	31.33 ± 0.14	58.82 ± 0.29	51.91 ± 0.38	**69.46** ± 0.14	55.13 ± 0.41	57.47 ± 0.12
	TaRA	64.77 ± 0.20	**67.28** ± 0.88	**65.52** ± 0.11	38.18 ± 0.16	**60.22** ± 0.70	54.52 ± 0.43	68.43 ± 0.16	**59.20** ± 0.49	**59.77** ± 0.36
LLaMA-2-7B	Full FT	70.62 ± 0.29	81.18 ± 0.09	78.04 ± 0.15	76.04 ± 0.09	79.03 ± 0.18	66.61 ± 0.04	83.71 ± 0.06	77.80 ± 0.00	76.63 ± 0.06
	LoRA	63.72 ± 0.21	77.24 ± 0.09	72.66 ± 0.16	56.50 ± 0.10	69.09 ± 0.16	61.66 ± 0.10	79.81 ± 0.16	64.33 ± 0.34	68.13 ± 0.01
	PiSSA	70.34 ± 0.15	80.92 ± 0.14	78.22 ± 0.05	85.83 ± 0.01	78.61 ± 0.07	**67.75** ± 0.07	**83.67** ± 0.06	77.73 ± 0.09	77.89 ± 0.01
	CorDA	70.42 ± 0.24	**80.94** ± 0.20	78.93 ± 0.09	83.43 ± 0.12	80.42 ± 0.34	66.95 ± 0.20	83.16 ± 0.07	77.60 ± 0.28	77.73 ± 0.02
	LoRA-One	71.22 ± 0.08	80.88 ± 0.08	78.48 ± 0.02	**89.19** ± 0.00	80.56 ± 0.07	67.12 ± 0.04	82.91 ± 0.04	79.40 ± 0.00	78.72 ± 0.02
	TaRA	**71.34** ± 0.04	80.49 ± 0.09	**79.00** ± 0.19	88.84 ± 0.03	**80.74** ± 0.11	66.98 ± 0.14	83.23 ± 0.08	**80.60** ± 0.28	**78.90** ± 0.04
LLaMA-3.1-8B	Full FT	74.39 ± 0.15	88.45 ± 0.07	80.35 ± 0.04	95.52 ± 0.00	85.98 ± 0.08	82.17 ± 0.07	92.31 ± 0.02	84.60 ± 0.16	85.47 ± 0.03
	LoRA	71.93 ± 0.15	85.91 ± 0.04	77.43 ± 0.07	92.63 ± 0.08	81.06 ± 0.11	78.24 ± 0.07	90.57 ± 0.06	79.47 ± 0.09	82.15 ± 0.02
	PiSSA	74.30 ± 0.23	88.21 ± 0.25	79.97 ± 0.08	95.19 ± 0.09	85.74 ± 0.30	80.03 ± 0.19	91.38 ± 0.44	86.13 ± 0.25	85.13 ± 0.02
	CorDA	74.56 ± 0.23	88.34 ± 0.23	80.16 ± 0.26	95.21 ± 0.02	85.37 ± 0.24	**80.81** ± 0.38	91.74 ± 0.07	**86.60** ± 0.16	85.35 ± 0.06
	LoRA-One	74.50 ± 0.05	88.03 ± 0.02	**80.34** ± 0.24	**96.04** ± 0.01	86.35 ± 0.43	80.66 ± 0.15	91.64 ± 0.07	86.33 ± 0.09	85.49 ± 0.06
	TaRA	**74.69** ± 0.27	**88.73** ± 0.06	80.33 ± 0.17	95.35 ± 0.05	**86.72** ± 0.32	80.49 ± 0.29	**91.86** ± 0.03	86.40 ± 0.16	**85.57** ± 0.04
Qwen-3-8B	Full FT	72.07 ± 0.09	89.74 ± 0.09	80.19 ± 0.13	93.19 ± 0.02	81.09 ± 0.20	91.86 ± 0.11	96.59 ± 0.06	88.87 ± 0.50	86.70 ± 0.06
	LoRA	71.39 ± 0.16	88.68 ± 0.13	79.60 ± 0.02	91.77 ± 0.01	77.66 ± 0.23	**91.53** ± 0.08	96.56 ± 0.02	88.13 ± 0.09	85.67 ± 0.04
	PiSSA	71.34 ± 0.30	89.88 ± 0.12	80.09 ± 0.18	94.25 ± 0.01	82.95 ± 0.13	91.44 ± 0.04	96.86 ± 0.04	91.07 ± 0.19	87.24 ± 0.03
	CorDA	71.19 ± 0.08	89.46 ± 0.10	81.19 ± 0.05	94.10 ± 0.01	82.35 ± 0.04	88.88 ± 0.08	95.62 ± 0.00	90.80 ± 0.00	86.70 ± 0.01
	LoRA-One	**73.26** ± 0.09	89.41 ± 0.14	80.16 ± 0.14	**94.57** ± 0.01	**84.95** ± 0.10	91.13 ± 0.07	**97.22** ± 0.00	90.40 ± 0.28	**87.64** ± 0.05
	TaRA	71.19 ± 0.17	**89.90** ± 0.07	**81.32** ± 0.04	94.38 ± 0.03	83.11 ± 0.07	89.85 ± 0.19	96.88 ± 0.09	**91.40** ± 0.00	87.25 ± 0.02

Table 2: Comparison of Full FT, LoRA, PiSSA, CorDA, LoRA-One, and TaRA on commonsense reasoning tasks. Results are averaged over three seeds, with standard deviations shown in gray text. Red and Orange color denote the best and second-best PEFT results in each model, excluding Full FT. Abbr. HS = HellaSwag, WG = WinoGrande.

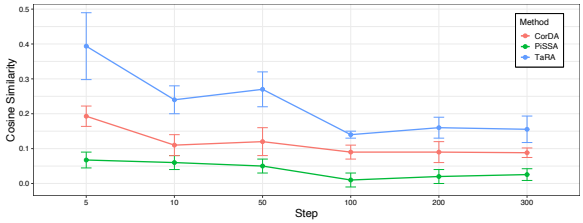


Figure 5: Gradient similarity, measured as the cosine similarity between the n-step (5, 10, 50, 100, 200 and 300) gradient of the combined LoRA adapter and that of the full weight, correlates with downstream accuracy.

them to the original parameter shape, and compute the corresponding one-step gradients. We then measure the cosine similarity between each low-rank gradient and the full-rank gradient to quantify how well each method preserves the original training-induced update direction.

We conduct this experiment on RoBERTa-base (Liu et al., 2019) trained for one step on the CoLA task (Warstadt et al., 2019) in the GLUE benchmark (Wang et al., 2018). Cosine similarities are computed for all linear layers over 100 random samples and averaged across layers.

As shown in Figure 1, TaRA achieves substantially higher alignment with the full-rank gradient than PiSSA and CorDA across all ranks. PiSSA shows little improvement as the rank increases, while CorDA exhibits only modest gains due to its task-aware initialization. In contrast, TaRA demonstrates a sharp increase in cosine similarity with rank and consistently attains the highest alignment, resulting in better fine-tuning quality. These results indicate that TaRA more effectively preserves training-relevant principal components that are critical for optimization.

Additionally, we further analyzed the effect of initialization beyond the very early stage of training. Figure 5 presents the gradient alignment measured after n training steps (5, 10, 50, 100, 200, and 300) under the same setup as the one-step analysis above. We observe that TaRA maintains substantially higher alignment than the other methods even after multiple training steps. We also found an interesting trend in this experiment: although the gradient similarity decreases from its initial level as training progresses, after a certain point it no longer drops and instead maintains a consistent level of similarity to Full FT. These results imply that, if the initialization induces gradients similar to those of Full FT at the beginning of training, its effect may persist throughout the training process.

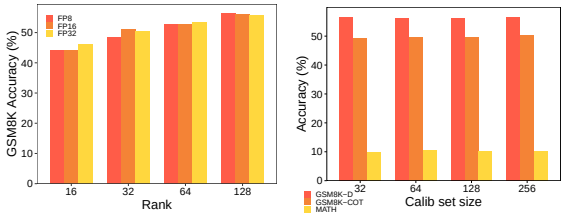
4.5 Initialization Overhead Analysis

Table 3 reports the initialization time, including covariance collection and SVD computation. For LLaMA-2-7B on MetaMathQA and CodeFeedback, initialization accounts for only 4-5% of the total fine-tuning time, demonstrating that its computational overhead is affordable in practice.

Since TaRA requires collecting both activation

($r = 128$)	Init Time	Training Time	
		LoRA	TaRA
MetaMathQA	18m	6h01m	6h20m (5%↑)
CodeFeedback	18m	6h40m	6h58m (4%↑)

Table 3: Initialization and training time comparison.



(a) init precision ablation (b) calib size ablation

Figure 6: (a) Ablation of the numeric precision used to collect covariance statistics for TaRA initialization (FP8/FP16/FP32), reporting GSM8K-D accuracy across ranks $r \in 16, 32, 64, 128$. (b) Ablation of the calibration set size (32, 64, 128, 256), reporting GSM8K-D accuracy.

and gradient covariances, its initialization can incur nontrivial memory overhead. To reduce this cost, we explore lower-precision accumulation of covariance statistics. Figure 6a shows the results in which covariances are collected in FP8, FP16, or FP32, followed by initialization and evaluation on LLaMA-2-7B for GSM8K-D across LoRA ranks. Lowering the precision results in only minor accuracy variations, with slight gains or losses depending on the rank. These results indicate that low-precision covariance collection is an effective alternative to reduce memory usage during initialization.

Additionally, we analyzed the performance variation with respect to the calibration set size. Figure 6b shows the performance trends across different calibration set sizes (32,64,128, and 256). The results show that TaRA achieves stable performance even with a small calibration set, with only minor variation in performance. This observation suggests that TaRA remains effective even when only a small calibration set is available.

5 Related Work

A wide range of PEFT methods have been proposed to reduce the cost of adapting large language models (Ding et al., 2023; Xu et al., 2023a). These methods span several paradigms, including adapter-based approaches (Houlsby et al., 2019; Lei et al., 2023; He et al., 2021; Rücklé et al., 2021; Zhao et al., 2022; Pfeiffer et al., 2021), prompt-based methods (Li and Liang, 2021; Hambardzumyan et al., 2021; Lester et al., 2021; Vu et al., 2022; Asai et al., 2022), and low-rank adaptation techniques (Hu et al., 2022; Zhang et al., 2023; Valipour et al., 2023). Adapter- and prompt-based methods introduce additional trainable components and optimize only these parameters, but they modify the model architecture and often incur inference overhead.

LoRA (Hu et al., 2022) addresses this limitation by freezing pretrained weights and learning a low-rank update parameterized by matrices $\mathbf{A}$ and $\mathbf{B}$, which project activations into and out of a bottleneck space. LoRA is simple, computationally efficient, and allows the learned update to be merged into the base weights after training, enabling inference without additional overhead.

Building on LoRA, many extensions have been developed to further improve efficiency and performance. These efforts primarily focus on allocating update capacity across layers and enabling training under resource constraints, particularly memory limitations. Representative approaches include adaptive rank allocation (Zhang et al., 2023; Valipour et al., 2023), redesigning low-rank update structures (Liu et al., 2024; Zhao et al., 2024; Qiu et al., 2023), and integrating LoRA with pruning (Zhang et al., 2024) or quantization (Dettmers et al., 2023; Xu et al., 2023b). Together, these methods mitigate the accuracy degradation caused by rank constraints while preserving LoRA's core benefit.

Notably, these advances are largely orthogonal to the choice of LoRA initialization. As a result, they are complementary to the method proposed in this work, and combining them may yield further performance gains.

6 Conclusion

In this paper, we propose training-aware low-rank adaptation initialization (TaRA), a new PEFT approach that explicitly prioritizes training-relevant directions. TaRA performs a covariance-aware SVD that jointly incorporates activation and gradient statistics to identify directions that are most influential during optimization. Using the resulting low-rank approximation, we show that the induced one-step gradient closely matches that of full fine-tuning, indicating effective preservation of training-relevant information. When applied to LoRA initialization, TaRA consistently outperforms existing methods and achieves state-of-the-art performance across a wide range of benchmarks.

7 Limitations

Our method, TaRA, is explicitly designed to optimize fine-tuning toward a given target task, and therefore it may show less stable generalization when evaluated in out-of-distribution settings. In the natural language generation setting, the code generation benchmarks HumanEval and MBPP are out-of-distribution with respect to the training dataset, CodeFeedback, since they differ in both input-output format and the underlying generation requirements. Under this distribution mismatch, Table 1 indicates that all compared methods, including LoRA, PiSSA, CorDA, LoRA-One, and TaRA, can display non-monotonic behavior as the rank changes, reflecting that increasing rank does not necessarily translate into steady gains under out-of-distribution evaluation. Nevertheless, this variability does not negate the advantages of TaRA in the setting it targets. On in-distribution tasks, TaRA consistently delivers strong accuracy and exhibits stable convergence behavior during training, suggesting that aligning initialization with training dynamics remains beneficial when the evaluation distribution matches the fine-tuning objective. These results highlight the effectiveness of training-aware initialization and offer a complementary perspective on LoRA design.

References

Akari Asai, Mohammadreza Salehi, Matthew E Peters, and Hannaneh Hajishirzi. 2022. Attempt: Parameter-efficient multi-task tuning via attentional mixtures of soft prompts. *arXiv preprint arXiv:2205.11961*.

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*.

Loubna Ben Allal, Niklas Muennighoff, Logesh Kumar Umapathi, Ben Lipkin, and Leandro von Werra. 2022. A framework for the evaluation of code generation models. https://github.com/bigcode-project/bigcode-evaluation-harness.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, and 1 others. 2024. Lessons from the trenches on reproducible evaluation of language models. *Preprint*, arXiv:2405.14782.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

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

Viktoriia Chekalina, Daniil Moskovskiy, Daria Cherniuk, Maxim Kurkin, Andrey Kuznetsov, and Evgeny Frolov. 2025. Generalized fisher-weighted svd: Scalable kronecker-factored fisher approximation for compressing large language models. *arXiv preprint arXiv:2505.17974*.

Mark Chen. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

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. *arXiv preprint arXiv:2110.14168*.

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

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.

Carl Eckart and Gale Young. 1936. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218.

Gene H Golub and Charles F Van Loan. 2013. *Matrix computations*. JHU press.

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*.

Guy Gur-Ari, Daniel A Roberts, and Ethan Dyer. 2018. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*.

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. Warp: Word-level adversarial reprogramming. *arXiv preprint arXiv:2101.00121*.

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

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*.

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*.

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.

Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. Language model compression with weighted low-rank factorization, 2022. *URL https://arxiv. org/abs/2207.00112*, 4.

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 Lee. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 5254–5276.

Ting Hua, Yen-Chang Hsu, Felicity Wang, Qian Lou, Yilin Shen, and Hongxia Jin. 2022. Numerical optimizations for weighted low-rank estimation on language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1404–1416.

Olivier Ledoit and Michael Wolf. 2012. Nonlinear shrinkage estimation of large-dimensional covariance matrices.

Tao Lei, Junwen Bai, Siddhartha Brahma, Joshua Ainslie, Kenton Lee, Yanqi Zhou, Nan Du, Vincent Zhao, Yuexin Wu, Bo Li, and 1 others. 2023. Conditional adapters: Parameter-efficient transfer learning with fast inference. *Advances in Neural Information Processing Systems*, 36:8152–8172.

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

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

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*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

James Martens. 2020. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76.

James Martens and Roger Grosse. 2015. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR.

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.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. Adapterfusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th conference of the European chapter of the association for computational linguistics: main volume*, pages 487–503.

Zeju Qiu, Weiyang Liu, Haiwen Feng, Yuxuan Xue, Yao Feng, Zhen Liu, Dan Zhang, Adrian Weller, and Bernhard Schölkopf. 2023. Controlling text-to-image diffusion by orthogonal finetuning. *Advances in Neural Information Processing Systems*, 36:79320–79362.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021. Adapterdrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 7930–7946.

Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. 2017. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

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.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2022. Spot: Better frozen model adaptation through soft prompt transfer. In *Proceedings of the 60th annual meeting of the association for computational linguistics*, pages 5039–5059.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pages 353–355.

Shaowen Wang, Linxi Yu, and Jian Li. 2024. Lora-ga: 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.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

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

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. 2023b. Qa-lora: Quantization-aware low-rank adaptation of large language models. *arXiv preprint arXiv:2309.14717*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

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.

Hong Yankun, Li Xing, Zhen Hui-Ling, Yu Xianzhi, Liu Wulong, and Yuan Mingxuan. 2025. Svdq: 1.25-bit and 410x key cache compression for llm attention. *arXiv preprint arXiv:2502.15304*.

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*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. 2024. Loraprune: Structured pruning meets low-rank parameter-efficient fine-tuning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3013–3026.

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

Yuanhe Zhang, Fanghui Liu, and Yudong Chen. 2025. Lora-one: One-step full gradient could suffice for fine-tuning large language models, provably and efficiently. *arXiv preprint arXiv:2502.01235*.

Hongyu Zhao, Hao Tan, and Hongyuan Mei. 2022. Tiny-attention adapter: Contexts are more important than the number of parameters. *arXiv preprint arXiv:2211.01979*.

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

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*.

A Proof of Equation

Second-order Taylor expansion. Let $\mathcal{L}$ be the loss and θ the parameter matrix of a model, with reference point θ_0. A second-order Taylor expansion of $\mathcal{L}(\theta)$ around θ_0 gives

$$\mathcal{L}(\theta) \approx \mathcal{L}(\theta_0) + \nabla\mathcal{L}(\theta_0)^\top(\theta - \theta_0) + \frac{1}{2}(\theta - \theta_0)^\top H(\theta - \theta_0), \qquad H := \nabla^2\mathcal{L}(\theta_0).$$

Hessian to Fisher curvature surrogate. Instead of explicitly forming the Hessian H, we use the Fisher information matrix $\mathcal{F}$ as a curvature surrogate, and replace the quadratic curvature term by $\mathcal{F}$:

$$\mathcal{L}(\theta) \approx \mathcal{L}(\theta_0) + \nabla\mathcal{L}(\theta_0)^\top(\theta - \theta_0) + \frac{1}{2}(\theta - \theta_0)^\top \mathcal{F}(\theta - \theta_0).$$

Fisher to K-FAC approximation. Following the K-FAC approximation (Martens and Grosse, 2015), we factorize the Fisher matrix as

$$\mathcal{F} \approx \Sigma_X \otimes \Sigma_G,$$

where Σ_X and Σ_G are the activation and gradient covariance matrices, respectively. Using standard Kronecker/vectorization identities, the quadratic form induced by $\mathcal{F}$ can be written as

$$(\theta - \theta_0)^\top(\Sigma_X \otimes \Sigma_G)(\theta - \theta_0) = \operatorname{tr}\Big((\theta - \theta_0)^\top \Sigma_G(\theta - \theta_0)\Sigma_X\Big).$$

Quadratic term. We define the corresponding curvature-induced quadratic term as

$$P(\theta) := \frac{1}{2}\operatorname{tr}\Big(\Sigma_G(\theta - \theta_0)\Sigma_X(\theta - \theta_0)^\top\Big).$$

Then the Taylor model can be rewritten as

$$\mathcal{L}(\theta) \approx \mathcal{L}(\theta_0) + \nabla\mathcal{L}(\theta_0)^\top(\theta - \theta_0) + P(\theta).$$

Differential/trace computation of $\nabla P(\theta)$. Let $E := \theta - \theta_0$. Then

$$P(\theta) = \frac{1}{2}\operatorname{tr}\Big(\Sigma_G E \Sigma_X E^\top\Big).$$

Taking the differential yields

$$dP(\theta) = \frac{1}{2}\operatorname{tr}\Big(\Sigma_G\, dE\, \Sigma_X\, E^\top\Big) + \frac{1}{2}\operatorname{tr}\Big(\Sigma_G\, E\, \Sigma_X\, dE^\top\Big).$$

Using cyclicity of trace and $\operatorname{tr}(A\, dE^\top) = \operatorname{tr}(A^\top dE)$, we obtain

$$dP(\theta) = \frac{1}{2}\operatorname{tr}\Big(\Sigma_X E^\top \Sigma_G\, dE\Big) + \frac{1}{2}\operatorname{tr}\Big((\Sigma_G E \Sigma_X)^\top dE\Big).$$

Since Σ_X and Σ_G are covariance matrices, we use symmetric estimates so that $\Sigma_X^\top = \Sigma_X$ and $\Sigma_G^\top = \Sigma_G$, which makes the two terms equal and yields

$$dP(\theta) = \operatorname{tr}\Big((\Sigma_G E \Sigma_X)^\top dE\Big).$$

Identifying the gradient and the final relation. By the defining relation for a scalar function,

$$dP(\theta) = \operatorname{tr}\Big((\nabla_E P(\theta))^\top dE\Big),$$

we identify

$$\nabla_E P(\theta) = \Sigma_G E \Sigma_X.$$

Finally, since $E = \theta - \theta_0$ implies $dE = d\theta$, we conclude

$$\nabla P(\theta) = \Sigma_G(\theta - \theta_0)\Sigma_X.$$

Taking the gradient of the Taylor model $\mathcal{L}(\theta) \approx \mathcal{L}(\theta_0) + \nabla\mathcal{L}(\theta_0)^\top(\theta - \theta_0) + P(\theta)$ gives

$$\nabla\mathcal{L}(\theta) \approx \nabla\mathcal{L}(\theta_0) + \nabla P(\theta),$$

and therefore

$$\nabla\mathcal{L}(\theta) - \nabla\mathcal{L}(\theta_0) \approx \Sigma_G(\theta - \theta_0)\Sigma_X.$$

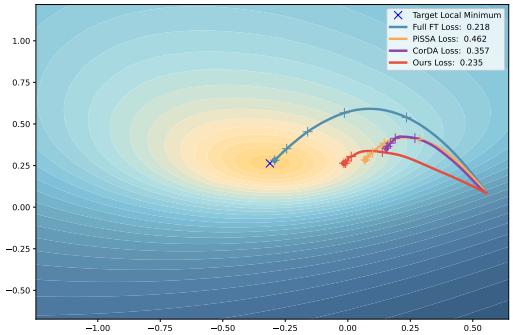


Figure 7: Loss landscape

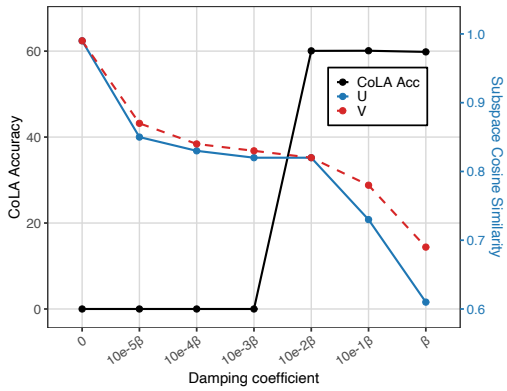


Figure 8: Damping coefficient ablation

Method	CoLA	SST-2	MRPC	STS-B	QNLI	AVG
Full FT	58.67	93.97	89.61	89.80	92.23	84.86
LoRA	54.68	94.31	87.25	88.73	92.52	83.50
PiSSA	58.46	94.22	88.17	89.64	92.55	84.61
CorDA	58.68	92.82	**89.07**	89.49	92.13	84.44
TaRA	**58.86**	**94.50**	87.50	**90.38**	**92.93**	**84.83**

Table 4: GLUE benchmark accuracy.

Calib Size	GSM8K-D	GSM8K-COT	MATH
32	**56.69 ± 0.22**	49.38 ± 0.11	9.88 ± 0.03
64	56.11 ± 0.33	49.55 ± 0.22	**10.41 ± 0.04**
128	56.05 ± 0.40	49.80 ± 0.18	10.23 ± 0.04
256	56.59 ± 0.25	**50.42 ± 0.13**	10.08 ± 0.10

Table 5: Ablation on calibration set size. We report mean and standard deviation over three seeds.

B Additional Analysis

This section presents additional analyses that could not be included in the main paper: (1) evaluation on the GLUE benchmark, and (2) a toy-example visualization of gradient trajectories illustrating the faster convergence of TaRA.

We fine-tune RoBERTa-base (Liu et al., 2019) on five GLUE tasks (Wang et al., 2018)—CoLA, SST-2, MRPC, STS-B, and QNLI—and compare Full Fine-Tuning, LoRA, PiSSA, CorDA, and TaRA. Table 4 reports the mean performance over five random seeds. TaRA achieves the best results on CoLA, SST-2, STS-B, and QNLI, and also yields the strongest average performance across the five tasks. These results indicate that TaRA remains effective for encoder-only language models, not only decoder-centric LLMs.

For all PEFT methods, we use LoRA with $r = \alpha = 128$, and train for 3 epochs with batch size 32.

Figure 7 visualizes the gradient trajectories of Full Fine-Tuning, PiSSA, CorDA, and TaRA in a toy setting. Full Fine-Tuning and PiSSA follow a noticeably curved trajectory, taking a relatively indirect route before converging toward a local minimum. CorDA exhibits a less circuitous early trajectory, suggesting improved initial alignment. In contrast, TaRA is initialized with a direction that heads more directly toward the local minimum and ultimately converges to a lower objective value than PiSSA and CorDA. This behavior is consistent with our hypothesis: by better preserving training-relevant components, TaRA facilitates faster and more effective optimization.

This toy experiment follows the setup of PiSSA (Meng et al., 2024). We pre-train a three-linear-layer network on 10,000 MNIST samples from the odd-number classes, and then fine-tune on 2,000 samples from the even-number classes. We use LoRA with rank and scaling set to $r = \alpha = 16$, and adopt a learning rate of 5×10^{-4}.

C Diagonal Damping Analysis

Figure 8 shows the results of our CoLA-task analysis on the effect of the diagonal damping coefficient c, including performance stabilization due to damping (left y-axis) and the change in the resulting SVD U and V spaces compared with the no-damping case (right y-axis). The results show that when c is smaller than 10^{-2}, including $c = 0$ (= no damping), task accuracy collapses to 0.0, indicating a failure to stabilize training and complete training breakdown. In contrast, when $c \geq 10^{-2}$, the performance becomes stable and training proceeds normally. Since 10^{-2} stabilizes the second-order information while preserving the original space in the no-damping case as much as possible, we choose 10^{-2} as an appropriate damping coefficient.

LoRA α	GSM8K-D	GSM8K-COT	MATH
64	54.59 ± 0.16	47.34 ± 0.14	9.98 ± 0.00
128	$\mathbf{56.59 \pm 0.25}$	$\mathbf{50.42 \pm 0.13}$	10.08 ± 0.10
256	55.04 ± 0.11	49.53 ± 0.20	$\mathbf{10.21 \pm 0.09}$

Table 6: Ablation on LoRA α. We report mean and standard deviation over three seeds.

D Calibration Set Size and LoRA alpha Ablation

Table 5 shows the performance variation across different calibration set sizes at rank 128. The results indicate that TaRA remains stable even with small calibration sets, with only minor performance changes. This suggests that TaRA can still produce meaningful results even when the calibration set must be kept small due to limited available data.

Table 6 shows the performance variation across different LoRA α values at rank 128. The results show that TaRA achieves the best performance when LoRA α is set equal to the LoRA rank.

E Difference between Fisher-based Model Compression and TaRA

Recent LLM compression methods perform compression using Fisher information (Hsu et al.; Hua et al., 2022; Chekalina et al., 2025). From the compression perspective, the following formulation is used:

$$\min_{\theta} \mathcal{L}(\theta) - \mathcal{L}(\theta_0) \approx \min_{\theta} \left\| \mathcal{F}^{1/2}(\theta - \theta_0) \right\|_F^2 .$$

It uses a formulation that finds θ by minimizing the change in the loss function $\mathcal{L}$. Specifically, it seeks θ that remains as close as possible to the original model θ_0, weighted by the square-root Fisher information matrix $F^{1/2}$.

In contrast, TaRA uses a formulation that finds θ by minimizing the change in the gradient:

$$\min_{\theta} \|\nabla\mathcal{L}(\theta) - \nabla\mathcal{L}(\theta_0)\| \approx \min_{\theta} \|\mathcal{F}(\theta - \theta_0)\| .$$

It uses a formulation that finds θ by keeping it as close as possible to the original model θ_0, weighted by the Fisher information matrix F without taking its square root.

Table 7 compares these two formulations for LoRA initialization on the MATH task at rank 128. The results show that applying LoRA initialization using the square-root formulation from the compression perspective leads to suboptimal performance. This supports the significance of the theoretical analysis of TaRA from the training perspective.

F Hyperparameters

This section documents the hyperparameters used in Section 4. Table 8 lists the hyperparameters for the natural language generation experiments, while Table 9 reports those used for the natural language understanding experiments.

Variant	GSM8K-D	GSM8K-COT	MATH
with sqrt (compression view)	53.12 ± 0.30	48.28 ± 0.19	8.97 ± 0.11
without sqrt (TaRA)	$\mathbf{56.59 \pm 0.25}$	$\mathbf{50.42 \pm 0.13}$	$\mathbf{10.08 \pm 0.10}$

Table 7: Ablation on the square-root scaling term. We report mean and standard deviation over three seeds.

Section	Hyperparameter	Value
LoRA	Rank r (= Scaling α)	32, 64, 128
	Dropout	0.0
Training	Epochs	1
	Learning rate (*except* LoRA-One)	4×10^{-5}
	Learning rate (for LoRA-One)	2×10^{-4}
	Batch size	8
	Gradient accumulation steps	16
	Seed	0, 1, 2
Model	Model name	`meta-llama/Llama-2-7b-hf`
	Max sequence length	512
Data	Dataset (for math)	`meta-math/MetaMathQA`
	Dataset (for code)	`m-a-p/CodeFeedback-Filtered-Instruction`
	Dataset split	`train[:100000]`
Optimization	Optimizer	`adamw_torch`
	Weight decay	0.0
	Warmup ratio	0.03
	LR scheduler	`cosine`

Table 8: Hyperparameters used for natural language generation tasks fine-tuning.

Section	Hyperparameter	Value
LoRA	Rank r (= Scaling α)	128
	Dropout	0.0
Training	Epochs	1
	Learning rate (*except* LoRA-One)	2×10^{-5}
	Learning rate (for LoRA-One)	2×10^{-4}
	Batch size	8
	Gradient accumulation steps	16
	Seed	0, 1, 2
Model	Model name 1	`deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B`
	Model name 2	`meta-llama/Llama-2-7b-hf`
	Model name 3	`meta-llama/Llama-3.1-8B`
	Model name 4	`Qwen/Qwen3-8B`
	Max sequence length	256
Data	Dataset	`zwhe99/commonsense_170k`
Optimization	Optimizer	`adamw_torch`
	Weight decay	0.0
	Warmup ratio	0.03
	LR scheduler	`cosine`

Table 9: Hyperparameters used for natural language understanding tasks fine-tuning.