CoLA: Collaborative Low-Rank Adaptation

Yiyun Zhou, Chang Yao, Jingyuan Chen[*]
Zhejiang University
{yiyunzhou, changy, jingyuanchen}@zju.edu.cn

Abstract

The scaling law of Large Language Models (LLMs) reveals a power-law relationship, showing diminishing return on performance as model scale increases. While training LLMs from scratch is resource-intensive, fine-tuning a pre-trained model for specific tasks has become a practical alternative. Full fine-tuning (FFT) achieves strong performance; however, it is computationally expensive and inefficient. Parameter-efficient fine-tuning (PEFT) methods, like LoRA, have been proposed to address these challenges by freezing the pre-trained model and adding lightweight task-specific modules. LoRA, in particular, has proven effective, but its application to multi-task scenarios is limited by interference between tasks. Recent approaches, such as Mixture-of-Experts (MOE) and asymmetric LoRA, have aimed to mitigate these issues but still struggle with sample scarcity and noise interference due to their fixed structure. In response, we propose CoLA, a more flexible LoRA architecture with an efficient initialization scheme, and introduces three collaborative strategies to enhance performance by better utilizing the quantitative relationships between matrices A and B. Our experiments demonstrate the effectiveness and robustness of CoLA, outperforming existing PEFT methods, especially in low-sample scenarios. Our data and code are fully publicly available[1].

1 Introduction

The scaling law (Kaplan et al., 2020; Zhai et al., 2022) of Large Language Models (LLMs) describes a power-law relationship between the performance of a deep learning model and its scale (*e.g.*, number of **parameters**, **computation**, and **data**) as the model size increases. As the model scale grows, the rate of performance improvement gradually flattens. Despite the impressive understanding and expressive capabilities of LLMs, training such a model from scratch is costly, which hinders the application of the scaling law, especially for small companies or institutions. Fine-tuning a single LLM for different downstream tasks or knowledge domains has become a common paradigm in various vertical fields (Araci, 2019; Peng et al., 2021; Chalkidis et al., 2020; Rasmy et al., 2021), and previous studies (Lester et al., 2021a; Hernandez et al., 2021) indicate that the scaling law also applies to fine-tuning. However, this approach, *i.e.*, full fine-tuning (FFT), requires entire pre-trained weights of the LLM to be involved in heavy gradient computation, demanding substantial computational resources and energy consumption, thus hindering further exploration of the scaling law. In response, parameter-efficient fine-tuning (PEFT) methods have been proposed, where the backbone model's parameters are frozen, and only a small number of additional parameters or external modules customized for specific tasks or multi-task learning are modified. Common methods include LoRA (Hu et al., 2021), Adapters (Lester et al., 2021b; Liu et al., 2021, 2022), and other variants (Liu et al., 2024b; Meng et al., 2024; Tian et al., 2024), which offer a solution for companies and researchers with limited computational resources.

As a PEFT method, LoRA has gained significant attention due to its simplicity, effectiveness, and generality, leading to many promising works, including exploration of better LoRA architectures. As shown in Figure 1 (a) and (b), unlike full fine-tuning, which completely unfreezes the pre-trained weight matrix W, LoRA freezes the pre-trained matrix W and approximates the incremental update ΔW of the pre-trained weights with two trainable low-rank matrices A and B. These matrices are inserted into each layer of the pre-trained model, achieving comparable or even superior performance to full fine-tuning. However, a single

[*]Corresponding author.
[1]`https://github.com/zyy-2001/CoLA`

LoRA module projects the features of different tasks into the same dense low-dimensional space, causing interference between tasks and failing to effectively separate the knowledge of different tasks, limiting adaptability in multi-task scenarios. Recent research (Feng et al., 2024; Liu et al., 2024a; Agiza et al., 2024) has introduced the Mixture-of-Experts (MOE) idea to decouple multi-task information in LoRA, as shown in Figure 1 (c). In this design, multiple experts are separately designed to learn task-shared and task-specific knowledge, with each expert consisting of a pair of low-rank matrices. This design allows the knowledge of multiple tasks to be effectively learned while maintaining parameter efficiency. Meanwhile, some studies (Tian et al., 2024; Yang et al., 2024a) have found that multiple smaller LoRA heads are more effective than a single LoRA, as matrix A tends to learn the commonality across all data, while matrix B focuses on the unique aspects of each intrinsic component. This leads to the design of an asymmetric LoRA architecture, as shown in Figure 1 (d). However, a single matrix A may struggle to capture commonality in limited samples and is prone to interference from noisy data. Moreover, these LoRA variant structures consistently initialize matrices A and B with Gaussian noise and zeros, which may result in small or random gradients early in training, slowing the fine-tuning process or causing the model to get stuck in suboptimal local minimum points. As a result, various alternative initialization schemes for LoRA have been explored (Hayou et al., 2024; Wang et al., 2024a; Meng et al., 2024), *e.g.*, PiSSA (Meng et al., 2024), which directly updates the model's principal components during fine-tuning through singular value decomposition, thereby accelerating convergence and improving performance.

Despite significant progress in the model parameter efficiency of different LoRA variant architectures, the scaling law for model fine-tuning remains limited in real-world scenarios due to the fact that sample labeling is expensive and private data is scarce. In particular, current LoRA methods, due to their fixed structure, fail to effectively capture the more complex inherent diversity in scarce samples. For example, as shown in Figure 3 in Sec. 4.2, LoRA's generalization ability deteriorates sharply when the sample size starts to decrease below 300. Therefore, it is crucial to further explore the numerical and collaborative relationships between matrices A and B in LoRA, freeing it from a monotonous structural design.

To address this, we introduce CoLA, a more flexible LoRA architecture, and extend the efficient PiSSA initialization scheme to CoLA, as shown in Figure 2 in Sec.3. CoLA does not enforce strict numerical relationships between matrices A and B. To fully leverage their collaborative potential, we have designed three collaborative strategies with different energy consumption (**computation**): (i) Fully collaborative CoLA$^{\intercal}$: Different matrices A and B interact and learn from each other, with deep parameter sharing. (ii) Random collaborative CoLA†: A single matrix is randomly selected, without relying on specific combinations, allowing more diverse parameter learning. (iii) Heuristic collaborative CoLA‡: A combination of the two structures, integrating the different advantages of LoRA structures. The details of these three collaborative strategies are described in Sec.3.3. We have conducted extensive experiments to validate the effectiveness and robustness of CoLA, with experiments performed on two recent Llama models with different **parameter** sizes, using fine-tuning **data** that reflect realistic scenarios. Four interesting observations are made, as presented in Sec. 4.2.

2 Related Works

2.1 LoRA Architecture

As a parameter-efficient fine-tuning (PEFT) method, LoRA has been widely used due to its simplicity, effectiveness, and generality. It effectively reduces the complexity of parameter updates by introducing low-rank decomposition, significantly improving fine-tuning efficiency while maintaining model performance. Previous research (Qin et al., 2021) has shown that despite the large number of parameters in pre-trained models, the intrinsic dimensionality of the model on downstream tasks is not large. Therefore, LoRA proposes the hypothesis of low-rank decomposition for the incremental update of pre-trained weight matrix $W_0 \in \mathbb{R}^{n\times m}$:

$$W = W_0 + \Delta W = W_0 + BA, \qquad (1)$$

where $B \in \mathbb{R}^{n\times r}, A \in \mathbb{R}^{r\times m}$, and $r \ll \min(n, m)$. It's noted that during the prediction phase, W_0, A and B can be combined into a single matrix without increasing the inference cost.

Due to its simplicity and practicality, the vanilla LoRA method has inspired considerable research. However, its simple structure sometimes struggles

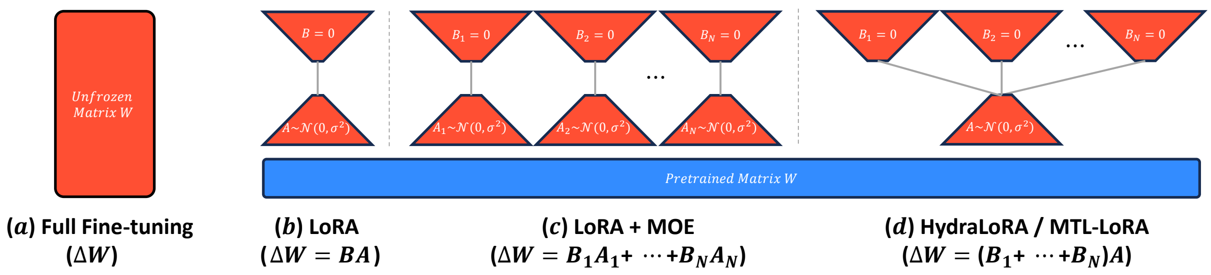

Figure 1: The comparison between Full Fine-tuning and different LoRA variant structures.

to effectively capture the diversity of data samples. As a result, some studies have attempted to combine the Mixture of Experts (MOE) approach, where different LoRA experts learn specific knowledge or tasks (Liu et al., 2024a; Feng et al., 2024; Agiza et al., 2024). In contrast, some methods are designed with the idea that the matrices A and B in LoRA tend to learn the commonality and intrinsic diversity of domain knowledge or tasks, leading to the proposal of asymmetric LoRA structures (Yang et al., 2024a; Tian et al., 2024). The idea of using two matrices to learn commonalities and diversities is consistent with the layer-wise abstraction mechanism (LeCun et al., 1998; Riesenhuber and Poggio, 1999; Hinton et al., 2006; Zhou et al., 2025a,b; Li et al., 2025) in deep learning. Inspired by this, a study (Gao et al., 2024) combining MOE and LoRA has found that higher layers should allocate more experts to effectively learn the more complex features. However, in these LoRA methods, the matrices A and B are limited by a fixed numerical relationship, and the collaborative relationship between the matrices has not been explored further, leading to suboptimal performance.

2.2 LoRA Parameter Initialization

Typically, LoRA initializes the matrices A and B with Gaussian noise and zeros, respectively, to enforce $\Delta W = 0$ at the beginning. Intuitively, initializing either the matrix A or the matrix B with zeros seems feasible, and empirical results (Zhu et al., 2024a) indicate that both approaches achieve similar performance. However, other research (Hayou et al., 2024) suggests that initializing the matrix B with zeros generally leads to better results. Meanwhile, other fixed initializations instead of random initialization have been explored (Meng et al., 2024; Ke et al., 2025). Specifically, PiSSA (Meng et al., 2024) shows significant performance improvements on several tasks by initializing with top singular vectors to accelerate the convergence of LoRA, and many similar works have been proposed afterward (Wang et al., 2024a,b; Yang et al., 2024b; Lingam et al., 2024; Zhang and Pilanci, 2024). Based on the effectiveness of singular value decomposition, we extend PiSSA to the proposed CoLA, and it serves as an essential component, as discussed in Observation 2 of Sec. 4.2.

3 CoLA

In this section, we introduce the proposed CoLA, a flexible LoRA achitecture as illustrated in Figure 2. We then present the extended PiSSA initialization scheme applied to CoLA and provide three collaborative strategies for the matrices A and B.

3.1 Flexible LoRA Architecture

Previous LoRA architectures have typically been constrained by fixed relationships between the number of matrices A and B. Specifically, as shown in Figures 1 (b) and (c), in vanilla LoRA and traditional LoRA + MOE architectures (Liu et al., 2024a), the setting is $\#A = \#B = N$, where $\#A$ and $\#B$ denote the number of matrices A and B, and N is a hyperparameter representing the number of experts. However, this symmetric structure may struggle to effectively learn both the shared components and intrinsic diversity of domain knowledge. To address this, some recent works have proposed asymmetric LoRA architectures. As illustrated in Figure 1 (d), HydraLoRA (Tian et al., 2024) and MTL-LoRA (Yang et al., 2024a) adopt the setting $\#A = 1, \#B = N$. However, a single matrix A may fail to capture commonalities in limited samples and is prone to data noise, especially in data-scarce real-world scenarios. In response, we introduce a more flexible LoRA architecture—CoLA—that frees itself from the fixed relationship between the number of matrices, *i.e.*, $\#A = M, \#B = N$, where M is also a hyperparameter, as illustrated in Figure 2. Notably, existing LoRA architectures can be viewed as special cases of CoLA.

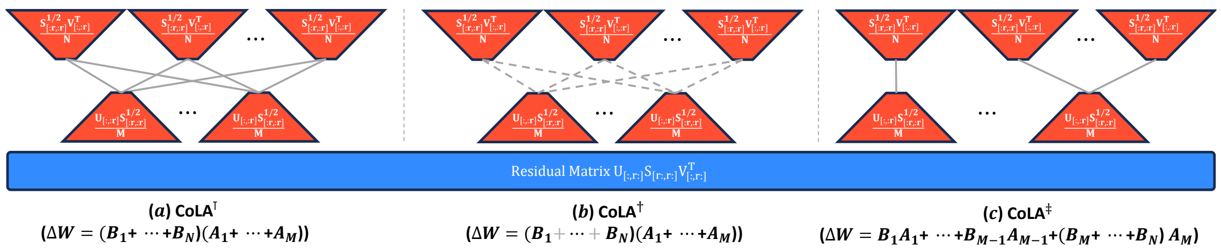
$\frac{S^{1/2}_{[:r,:r]}V^T_{[:,:r]}}{N}$
$\frac{U_{[:,:r]}S^{1/2}_{[:r,:r]}}{M}$
Residual Matrix $U_{[:,r:]}S_{[r:,r:]}V^T_{[:,r:]}$
(a) CoLA⊺
$(\Delta W = (B_1 + \cdots + B_N)(A_1 + \cdots + A_M))$
(b) CoLA†
$(\Delta W = (B_1 + \cdots + B_N)(A_1 + \cdots + A_M))$
(c) CoLA‡
$(\Delta W = B_1A_1 + \cdots + B_{M-1}A_{M-1} + (B_M + \cdots + B_N)A_M)$

Figure 2: Overview of CoLA with three collaborative strategies.

3.2 Extended PiSSA

For any matrix $W \in \mathbb{R}^{n\times m}$, a singular value decomposition (SVD) of the following form can be found:

$$W = USV^\top, \tag{2}$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices, and $V^\top$ is the transpose of V. $S \in \mathbb{R}^{n\times m}$ is a non-negative diagonal matrix:

$$S_{i,j} = \begin{cases} \Lambda_i, & i = j \\ 0, & i \neq j \end{cases} \tag{3}$$

where the diagonal elements are in descending order, *i.e.*, $\Lambda_1 \geq \Lambda_2 \geq \cdots \geq 0$, known as the singular values.

PiSSA divides S, along with U and V, into two groups: the principal singular values and vectors $(\{U_{[:,:r]}, S_{[:r,:r]}, V_{[;,:r]}\})$ and the residual singular values and vectors $(\{U_{[:,r:]}, S_{[r:,r:]}, V_{[;,r:]}\})$. The principal singular values and vectors are used to initialize the matrices A and B in LoRA:

$$A = U_{[:,:r]}S^{1/2}_{[:r,:r]}, \quad B = S^{1/2}_{[:r,:r]}V^\top_{[:,:r]}. \tag{4}$$

Meanwhile, the residual singular values and vectors are used to construct the residual matrix, which remains frozen during fine-tuning:

$$W_0 = U_{[:,r:]}S_{[r:,r:]}V^\top_{[:,r:]}. \tag{5}$$

Since elements of $\Lambda_{[:r]} \gg$ elements of $\Lambda_{[r:]}$, PiSSA assumes that the initial BA contains the most important directions of W, leading to faster and better convergence. This assumption is supported by the Eckart-Young-Mirsky theorem, which is ignored by PiSSA, as formally described in Theorem 3.1.

Theorem 3.1 *If the SVD of $W \in \mathbb{R}^{n\times m}$ is $USV^\top$, then the optimal rank r approximation of W is $U_{[:n,:r]}S_{[:r,:r]}V^\top_{[:m,:r]}$.*

We extend PiSSA to the flexible CoLA architecture. For the principal singular values and vectors of matrices A and B, $U_{[:,:r]}S^{1/2}_{[:r,:r]}$ and $S^{1/2}_{[:r,:r]}V^T_{[:,:r]}$, which are evenly distributed to each matrix A_i $(1 \leq i \leq M)$ and B_j $(1 \leq j \leq N)$:

$$A_i = \frac{U_{[:,:r]}S^{1/2}_{[:r,:r]}}{M}, \quad B_j = \frac{S^{1/2}_{[:r,:r]}V^T_{[:,:r]}}{N}. \tag{6}$$

Each matrix A_i and B_j is initially treated equally and aligned for full fine-tuning. During the fine-tuning process, each matrix is optimized in different directions, which allows CoLA to have more diverse generalization capabilities. The extended PiSSA initialization significantly benefits CoLA, as observed in Observation 2 of Sec. 4.2.

3.3 Collaborative Strategy

In the vanilla LoRA method, the matrices A and B represent the incremental update $\Delta W = BA$ of pre-trained weights through a one-to-one relationship. However, this simple, singular connection fails to capture the diversity of different tasks. Therefore, some LoRA methods introduce the idea of MOE (Mixture of Experts) to construct multiple distinct one-to-one relationships within the matrices A and B to achieve the more diverse incremental update $\Delta W = B_1A_1 + \cdots + B_NA_N$. However, since each expert is independent, although it can effectively learn the intrinsic diversity of knowledge, it struggles to capture the commonality of domain-specific knowledge. Thus, HydraLoRA and MTL-LoRA methods adopt a one-to-many relationship between the matrices A and B to represent the more complex incremental update $\Delta W = (B_1 + \cdots + B_N)A$. However, the knowledge that a single A matrix can learn is limited and more prone to noise, especially in real-world scenarios with sparse samples. Based on this, we introduce a many-to-many relationship within the matrices A and B to represent the more refined incremental update, as shown in Figure 2. The three collaborative strategies are as follows:

Domain	Fine-tuning Dataset	Benchmark	Function
Generality	databricks-dolly-15k (Conover et al., 2023)	MMLU (Hendrycks et al., 2020)	General Instruction Following
Law	Lawyer-Instruct (Alignment-Lab-AI, 2024) and US-Terms (Chalkidis et al., 2023)	Legal Tasks in MMLU	Legal Judgment
Medicine	GenMedGPT-5k and clinic-10k from ChatDoctor (Li et al., 2023)	Medical Tasks in MMLU	Medical Diagnosis
Math	Training Set of GSM8k (Cobbe et al., 2021)	Test Set of GSM8k	Mathematical Reasoning
Finance	Training Set of fingpt-fineval (Wang et al., 2023)	Test Set of fingpt-fineval	Financial Q&A
Multi-tasking	OpenOrca (Lian et al., 2023)	Big-Bench Hard (BBH) (Suzgun et al., 2022)	Natural Language Understanding (NLU) and Natural Language Generation (NLG)

Table 1: The basic information of the datasets used in our experiments.

- **Fully Collaborative CoLA$^\top$**: CoLA$^\top$ represents the finest incremental update $\Delta W = (B_1 + \cdots + B_N)(A_1 + \cdots + A_M)$ by combining each matrix A and B. This collaborative strategy breaks down the information transmission barrier between each matrix A and B, making it easier for beneficial knowledge to be shared. However, this may introduce more energy consumption.

- **Random Collaborative CoLA†**: Inspired by the dropout regularization technique (Srivastava et al., 2014) in deep learning, CoLA† represents the more robust incremental update $\Delta W = (B_1 + \cdots + B_N)(A_1 + \cdots + A_M)$ by combining each matrix A with a randomly chosen matrix B. This collaborative strategy does not rely on a specific combination, making the learned knowledge expected to be more robust, while incurring the fewest energy consumption.

- **Heuristic Collaborative CoLA‡**: CoLA‡ integrates multiple one-to-one and one-to-many relationships between matrices A and B (which can be seen as the combination of Figures 1 (c) and (d)) to represent the complex and diverse incremental update $\Delta W = B_1A_1 + \cdots + B_{M-1}A_{M-1} + (B_M + \cdots + B_N)A_M$ (assume $M < N$). This collaborative strategy combines the advantages of two specific combinations, enabling the learning of both general and diverse knowledge, while producing moderate energy consumption.

We analyze the energy consumption produced by these three collaborative strategies in Observation 4 of Sec. 4.2.

4 Experiments

In this section, we present the setup and details of the experiments. Then, we share our findings and provide concise explanations.

4.1 Experimental Setup

4.1.1 Datasets and Benchmarks

Following HydraLoRA (Tian et al., 2024), we evaluate the performance of different fine-tuning methods on datasets from single and multiple domains. Table 1 shows the basic information of these datasets, and more details can be found in Appendix A.

4.1.2 Baselines

We select recent Llama models with different parameter scales (Dubey et al., 2024) (Llama-3.2-3B and Llama-3.1-8B) as the backbone models for optimization. Additionally, we compare CoLA with several different PEFT methods:

- Single domain: Full fine-tuning (FFT) (not applied to Llama-3.1-8B due to resource limitations), Prompt Tuning (Lester et al., 2021b), P-Tuning (Liu et al., 2021), IA^3 (Liu et al., 2022), LoRA (Hu et al., 2021), DoRA (Liu et al., 2024b), PiSSA (Meng et al., 2024), HydraLoRA (Tian et al., 2024).

- Multiple domain: MOELoRA (Liu et al., 2024a), MTL-LoRA (Yang et al., 2024a), MoLA (Gao et al., 2024), HydraLoRA.

Details of the aforementioned PEFT methods can be found in Appendix B. Specifically, unless stated otherwise, the LoRA rank is set to 8 by default for the PEFT methods related to LoRA.

4.1.3 Implementation

Previous studies have shown that the quality of text generated by models can be significantly influenced by sampling strategies (*e.g.*, temperature), and that generating the entire text leads to inefficiencies (Renze and Guven, 2024; Zhu et al., 2024b; Patel et al., 2024). Meanwhile, the multiple-choice inference mode generates stable and consistent results by using the model's logits, requiring only the computation of the logits for the final token (Hendrycks et al., 2021). The specific differences between these two inference modes, as well as the code, can be found in Appendix C. To

Method	Llama-3.1-8B	Prompt Tuning	P-Tuning	IA^3	LoRA$_{r=8}$	LoRA$_{r=16}$	LoRA$_{r=24}$	LoRA$_{r=32}$	DoRA	PiSSA	HydraLoRA	CoLA	CoLA$^\intercal$	CoLA†	CoLA‡
#A | #B	-	-	-	-	1 | 1	1 | 1	1 | 1	1 | 1	1 | 1	1 | 1	1 | 3	1 | 3	2 | 3	2 | 3	2 | 3
%Param	-	0.0004	0.0280	0.0065	0.2605	0.5196	0.7774	1.0338	0.2605	0.2605	0.5785	0.5325	0.6551	0.6551	0.6551
Generality	22.95	25.38	24.68	22.96	50.36	51.47	52.91	52.35	51.56	54.72	45.86	58.04**	58.21**	42.26	57.08**
Law	24.62	25.75	26.09	24.62	25.98	26.60	24.96	25.92	26.32	26.58	26.26	36.25**	41.46**	26.27	31.04*
Medicine	23.82	24.91	25.12	23.82	42.66	47.17	50.38	45.94	43.28	44.64	40.61	56.11**	54.33**	41.24	50.23
Math	24.79	26.00	26.16	24.72	51.02	54.66	56.94	56.48	51.18	57.00	47.31	57.71	59.14**	45.96	56.34
Finance	26.42	28.30	21.51	26.42	40.38	44.53	45.66	48.68	41.13	46.79	38.87	52.45**	50.19**	37.51	45.32

Table 2: Comparison of 0-shot performance (%) of different fine-tuning methods based on Llama-3.1-8B across multiple single domains. The experiments are repeated 5 times under random seeds 42 to 46 and the average performance is reported. #A and #B represent the number of matrices A and B, respectively. * and ** indicate that the improvements over the strongest baseline with underlined are statistically significant, with p <0.05 and p <0.01, respectively. The results based on Llama-3.2-3B are in Appendix E.

Method		Base	LoRA$_{r=64}$	MOELoRA	MTL-LoRA	MoLA	HydraLoRA	CoLA	CoLA$^\intercal$	CoLA†	CoLA‡
#A | #B		-	1 | 1	8 | 8	1 | 14	$\tilde{8}$ | $\tilde{8}$	1 | 14	1 | 14	4 | 10	4 | 10	4 | 10
%Param		-	2.0465	2.0482	2.0051	2.1654	2.2107	2.0026	1.8330	1.8330	1.8330
Multi-tasking	**Llama-3.2-3B**	29.36	34.89	30.77	32.31	35.11	29.64	36.87**	36.47*	31.18	34.58
	Llama-3.1-8B	29.47	42.99	40.53	41.39	41.16	39.08	42.87	43.62*	39.26	42.16

Table 3: Comparison of 0-shot performance (%) of different fine-tuning methods across multiple domains.

ensure fairness and reproducibility in evaluation, we follow the evaluation settings recommended in LlamaFactory (Zheng et al., 2024) and uniformly convert the model's generation task into a classification task. However, this may conflict with the original instruction setup. We use powerful language models to normalize the conflicting datasets (GSM8K and BBH) into multiple-choice formats. All experiments are conducted using the LlamaFactory framework. The prompts for normalization, the prompt template used for the model's zero-shot evaluation, and additional experimental details can be found in Appendix D.

4.2 Results

We conduct extensive experiments to demonstrate the value of the proposed CoLA. Four key observations are summarized as follows.

Observation 1. CoLA is effective on both single and multiple knowledge domains.

Table 2 and Table 5 in Appendix E show the performance comparison of different fine-tuning methods based on Llama-3.1-8B and Llama-3.2-3B across multiple single domains in a 0-shot setting. Our findings are as follows: (1) Compared to methods like LoRA, fine-tuning approaches including Prompt Tuning, P-Tuning, and IA^3, although designed with fewer parameters, struggle to effectively capture the patterns of few samples and generalize to more samples in scenarios where data is scarce, which can become a curse in practical settings. (2) LoRA remains a simple yet hard-to-beat baseline compared to the base model and other methods, consistently achieving top-2 performance across these baselines. (3) DoRA generally improves the performance of LoRA$_{r=8}$, indicating the effectiveness of updating the weight decomposition's directional components in LoRA. (4) PiSSA achieves impressive performance with fewer parameters, due to its ability to pre-learn the principal components of the pre-trained weights for faster convergence. This also confirms the correctness of CoLA's efficient initialization scheme. Meanwhile, HydraLoRA does not achieve the expected results, likely due to its Gaussian noise initialization, which may lead to overfitting under conditions with scarce samples. (5) CoLA, CoLA$^\intercal$, and CoLA‡ (especially CoLA and CoLA$^\intercal$) consistently achieve excellent performance, which is closely tied to the advantages analyzed in Sec. 3.3, while CoLA† does not yield the expected results. This pseudo-LoRA implementation, averaging a matrix B, contradicts the quantitative relationship between the matrices A and B identified in Observation 3.

Table 3 presents a performance comparison of different fine-tuning methods across multiple domains in a 0-shot setting. Several insights are drawn: (1) Compared to the base model, the three multi-task LoRA method (MOELoRA, MTL-LoRA, and HydraLoRA) are effective, but fail to effectively learn the instruction patterns in the pre-trained weights due to the random initialization of their experts. (2) MoLA (specifically MoLA-∇) and LoRA$_{r=64}$ show comparable performance and consistently outperform MOELoRA, indicating the effectiveness of allocating more experts to higher layers with advanced features. (3) With the similar parameters, CoLA and CoLA$^\intercal$ consistently achieve the best performance, which indicates that they can

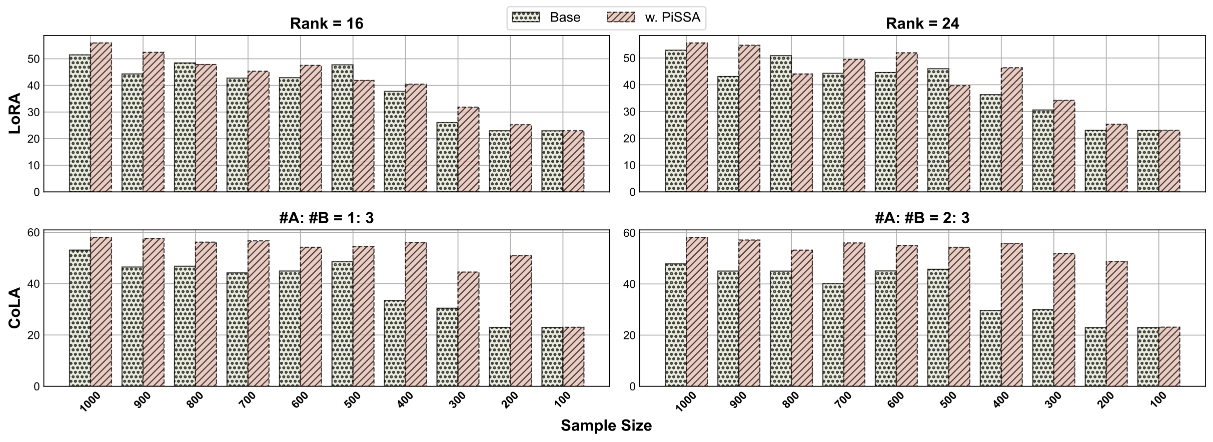

Figure 3: The impact of PiSSA initialization on LoRA and CoLA based on Llama-3.1-8B in the generality domain when the sample size is reduced.

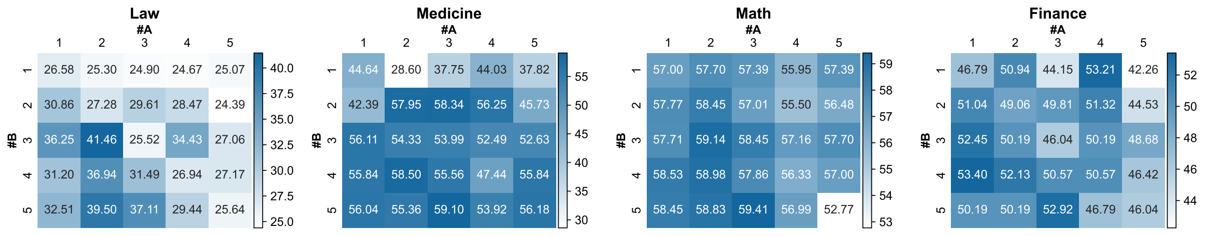

Figure 4: The performance of Llama-3.1-8B when the number of matrices A and B in CoLA differs across the domains of law, medicine, math, and finance.

also learn the inherent diversity of domain-specific knowledge in multitask learning. This is due to the fully collaborative strategy, which allows each matrix A and B to fully share knowledge.

> **Observation 2.** Compared to the LoRA method, the impact of parameter initialization on CoLA is more significant when the sample size is reduced.

We investigate the robustness of the CoLA initialization scheme with continuously decreasing samples in harsh environments. In the generality domain, based on Llama-3.1-8B, we select LoRA$_{r=16}$ and LoRA$_{r=24}$ as the baselines for our CoLA ($\#A = 1, \#B = 3$) and CoLA$^{\intercal}$ ($\#A = 2, \#B = 3$), respectively, to explore the necessity of the extended PiSSA initialization for CoLA, as shown in Figure 3. From this, we observe: (1) In extremely harsh environments (sample size = 100), all methods fail, especially the LoRA, whose performance starts to degrade sharply when the sample size reaches 300. (2) CoLA without the extended PiSSA, while not outperforming LoRA, demonstrates consistently the best performance after initialization and remains stable even with fewer samples (sample size = 200). This suggests that the extended PiSSA has a significant impact on CoLA. CoLA's ability to maintain high performance in such challenging environments is attributed to its use of multiple different matrices A and B, and the extended PiSSA initialization allows both matrices to learn the foundational instruction patterns of the pre-trained model, enabling efficient learning in distinct directions, leading to improved generalization.

> **Observation 3.** In CoLA, the number of matrix A should be fewer than the number of matrix B, as the benefit of increasing the matrix B outweighs that of increasing the matrix A.

We explore the quantitative relationship between matrices A and B in CoLA across four domains: law, medicine, math, and finance. The experiments are based on Llama-3.1-8B, where $\#A$ and $\#B$ range from 1 to 5. From Figure 4, we can observe:

- When either matrix A or B is fixed, increasing the number of the other matrix generally benefits the model. However, excessive increase can lead to overfitting (*e.g.*, $\#A = 5, \#B = 1$ in the finance domain), highlighting the need for careful tuning of the number of A and B matrices.

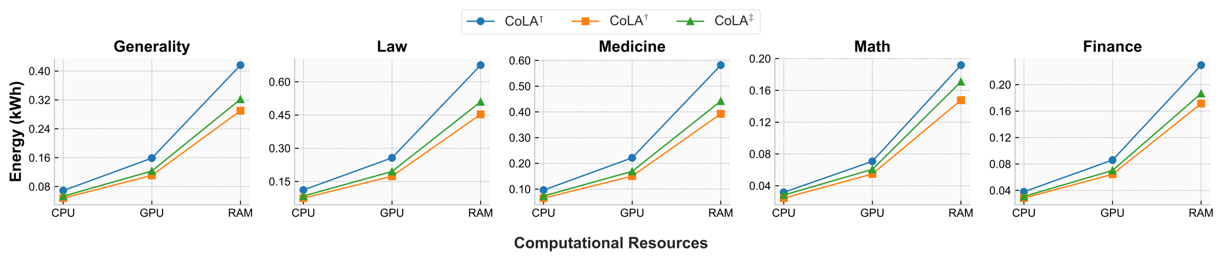

Figure 5: Energy consumption of three collaborative strategies based on Llama-3.1-8B.

- When $\#A = \#B$, simply increasing the number of experts does not necessarily improve model performance and may even degrade it (*e.g.*, $\#A = \#B = 5$ in the math domain). This suggests that traditional LoRA + MOE architectures, like MOELoRA, may not be optimal. On the contrary, when matrices A and B are asymmetric, particularly when $\#A < \#B$, increasing the number of experts tends to benefit the model (*e.g.*, $\#A = 1, \#B = 3 \rightarrow \#A = 2, \#B = 4 \rightarrow \#A = 3, \#B = 5$).

- Let $x < y$. The model benefits more from $\#A = x, \#B = y$ rather than $\#A = y, \#B = x$. That is, the number of matrix A in CoLA should be fewer than that of matrix B. This insight is inspired by the model structure designs in HydraLoRA and MoLA-∇: matrix A learns the underlying commonalities in the data, while matrix B focuses on the unique aspects of each component, with higher-level features receiving more weight. This can also be drawn from real life, where people often remember the contour of a face but overlook facial details (*e.g.*, the width of the nose), though these details are crucial for accurate facial recognition (Zhao et al., 2003; Liu and Liu, 2010; Tan and Triggs, 2010).

Observation 4. The three collaborative strategies (CoLA$^{\top}$, CoLA† and CoLA‡) have significantly different energy consumption, and more refined collaborative strategies are worth exploring.

We analyze the energy consumption of three collaborative strategies in CoLA (CoLA$^{\top}$, CoLA†, and CoLA‡) to validate the low energy consumption of our experimental setup. Our experiments are conducted on a GPU infrastructure powered by two NVIDIA A800 80GB GPUs and an Intel(R) Xeon(R) Platinum 8358 CPU @ 2.60GHz. We use CodeCarbon (Patterson et al., 2021) to record the energy consumption of CoLA$^{\top}$, CoLA†, and CoLA‡ based on Llama-3.1-8B across different single domains, with a random seed of 42. As shown in Figure 5, we observe the following: (1) The energy consumption of the three collaborative strategies—CoLA$^{\top}$, CoLA†, and CoLA‡—differs significantly, representing high, low, and medium configurations, respectively, which are suitable for different real-world application scenarios to meet diverse user needs. (2) Compared to the energy consumption results from HydraLoRA (Tian et al., 2024), our experiment consumes less than 1/10th of the energy. This is not only due to our sampling a smaller number of samples, which aligns with the scarcity of samples in real-world scenarios and presents a significant challenge for the performance of current PEFT methods, but also due to our experimental setup. We convert the model's generation evaluation into a classification evaluation, resulting in fewer tokens (the model's output is often just a capital letter). More details on our evaluation method can be found in Appendix D.

Domain	**Generality**	**Law**	**Medicine**	**Math**	**Finance**	**Multi-tasking**
CoLA†	42.26	26.27	41.24	45.96	37.51	39.26
CoLA$^{\widehat{\dagger}}$	52.26	31.02	48.15	54.89	46.32	40.64

Table 4: Performance comparison of two different random collaborative strategies based on Llama-3.1-8B.

However, we also observe that the CoLA† collaborative strategy performs poorly overall in our experiments, as it violates the principle identified in Observation 3. In contrast, we introduce another random collaborative strategy, CoLA$^{\widehat{\dagger}}$, which combines each matrix B with a random matrix A. We fine-tune Llama-3.1-8B on multiple different single-domain and multi-domain tasks, as shown in Table 4. From the table, we find that CoLA$^{\widehat{\dagger}}$ consistently outperforms CoLA†, which demonstrates the universality and effectiveness of Observation 3. Furthermore, due to space limitations, more refined collaborative strategies that align with the principles we discover remain to be explored. For example, as shown in Figure 2, where matrix A and B form a bipartite graph, this graph has many interesting properties (*e.g.*, maximum matching), making it a promising avenue for future research.

5 Conclusion

In conclusion, this paper introduces CoLA, a flexible LoRA architecture designed to address the limitations of current parameter-efficient fine-tuning methods, particularly in scenarios with scarce data. By decoupling the rigid numerical relationship between matrices A and B, CoLA enables more effective collaboration through three various strategies. Through extensive experimentation on multiple Llama models, we demonstrate that CoLA significantly improves generalization and robustness, especially in resource-constrained environments.

6 Limitations

Our experiment involves multiple knowledge domains: generality, law, medicine, math, finance, and mixed domains (multi-tasking). However, we do not validate the proposed CoLA method in the code domain, which is related to the consistent evaluation approach we adopted (also recommended by the LlamaFactory framework), as shown in Appendix D. Additionally, code-related questions are challenging to transform into multiple-choice questions. We are actively seeking a simple yet nontrivial way to add to the model's input tokens. Furthermore, the more refined collaborative strategy in the CoLA method is worth exploring. For example, as shown in Figure 2, a bipartite graph is formed between matrices A and B. Investigating the maximum matching of such a bipartite graph and identifying appropriate scenarios is promising, and we leave this as future work.

Acknowledgements

This work was supported by the "Pioneer" and "Leading Goose" R&D Program of Zhejiang under Grant No. 2025C02022.

References

Ahmed Agiza, Marina Neseem, and Sherief Reda. 2024. Mtlora: Low-rank adaptation approach for efficient multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16196–16205.

Alignment-Lab-AI. 2024. Lawyer-instruct.

D Araci. 2019. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*.

BIG bench authors. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. Legal-bert: The muppets straight out of law school. *arXiv preprint arXiv:2010.02559*.

Ilias Chalkidis, Nicolas Garneau, Catalina Goanta, Daniel Katz, and Anders Søgaard. 2023. LeXFiles and LegalLAMA: Facilitating English multinational legal language model development. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15513–15535, Toronto, Canada. Association for Computational Linguistics.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm. *Company Blog of Databricks*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Yu Han, and Hao Wang. 2024. Mixture-of-loras: An efficient multitask tuning for large language models. *arXiv preprint arXiv:2403.03432*.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Baochen Sun, Ruibo Liu, Daiyi Peng, Yawen Zhang, Xiaoyuan Guo, Jie Yang, and VS Subrahmanian. 2024. Higher layers need more lora experts. *arXiv preprint arXiv:2402.08562*.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. The impact of initialization on lora finetuning dynamics. *arXiv preprint arXiv:2406.08447*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. 2021. Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Wenjun Ke, Jiahao Wang, Peng Wang, Jiajun Liu, Dong Nie, Guozheng Li, and Yining Li. 2025. Unveiling lora intrinsic ranks via salience analysis. *Advances in Neural Information Processing Systems*, 37:131575–131595.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

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

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021b. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3045–3059. Association for Computational Linguistics.

Kunxi Li, Tianyu Zhan, Kairui Fu, Shengyu Zhang, Kun Kuang, Jiwei Li, Zhou Zhao, Fan Wu, and Fei Wu. 2025. Mergenet: Knowledge migration across heterogeneous models, tasks, and modalities. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 4824–4832.

Yunxiang Li, Zihan Li, Kai Zhang, Ruilong Dan, Steve Jiang, and You Zhang. 2023. Chatdoctor: A medical chat model fine-tuned on a large language model meta-ai (llama) using medical domain knowledge. *Cureus*, 15(6).

W Lian, B Goodson, E Pentland, et al. 2023. Openorca: An open dataset of gpt augmented flan reasoning traces.

Vijay Lingam, Atula Tejaswi, Aditya Vavre, Aneesh Shetty, Gautham Krishna Gudur, Joydeep Ghosh, Alex Dimakis, Eunsol Choi, Aleksandar Bojchevski, and Sujay Sanghavi. 2024. Svft: Parameter-efficient fine-tuning with singular vectors. *arXiv preprint arXiv:2405.19597*.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. 2024a. When moe meets llms: Parameter efficient fine-tuning for multi-task medical applications. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1104–1114.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024b. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT understands, too. *CoRR*, abs/2103.10385.

Zhiming Liu and Chengjun Liu. 2010. Fusion of color, local spatial and global frequency information for face recognition. *Pattern Recognition*, 43(8):2882–2890.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, pages 22631–22648. PMLR.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. Pissa: Principal singular values and singular vectors adaptation of large language models. *arXiv preprint arXiv:2404.02948*.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*.

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

Dhavalkumar Patel, Prem Timsina, Ganesh Raut, Robert Freeman, Matthew A levin, Girish N Nadkarni, Benjamin S Glicksberg, and Eyal Klang. 2024. Exploring temperature effects on large language models across various clinical tasks. *medRxiv*, pages 2024–07.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*.

Branislav Pecher, Ivan Srba, and Maria Bielikova. 2024. Fine-tuning, prompting, in-context learning and instruction-tuning: How many labelled samples do we need? *arXiv preprint arXiv:2402.12819*.

Shuai Peng, Ke Yuan, Liangcai Gao, and Zhi Tang. 2021. Mathbert: A pre-trained model for mathematical formula understanding. *arXiv preprint arXiv:2105.00377*.

Yujia Qin, Xiaozhi Wang, Yusheng Su, Yankai Lin, Ning Ding, Jing Yi, Weize Chen, Zhiyuan Liu, Juanzi Li, Lei Hou, et al. 2021. Exploring universal intrinsic task subspace via prompt tuning. *arXiv preprint arXiv:2110.07867*.

Laila Rasmy, Yang Xiang, Ziqian Xie, Cui Tao, and Degui Zhi. 2021. Med-bert: pretrained contextualized embeddings on large-scale structured electronic health records for disease prediction. *NPJ digital medicine*, 4(1):86.

Matthew Renze and Erhan Guven. 2024. The effect of sampling temperature on problem solving in large language models. *arXiv preprint arXiv:2402.05201*.

Maximilian Riesenhuber and Tomaso Poggio. 1999. Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019–1025.

Raphael Schäfer, Till Nicke, Henning Höfener, Annkristin Lange, Dorit Merhof, Friedrich Feuerhake, Volkmar Schulz, Johannes Lotz, and Fabian Kiessling. 2024. Overcoming data scarcity in biomedical imaging with a foundational multi-task model. *Nature Computational Science*, 4(7):495–509.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Xiaoyang Tan and Bill Triggs. 2010. Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE transactions on image processing*, 19(6):1635–1650.

Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Chengzhong Xu. 2024. Hydralora: An asymmetric lora architecture for efficient fine-tuning. *arXiv preprint arXiv:2404.19245*.

Hoang Van. 2023. Mitigating data scarcity for large language models. *arXiv preprint arXiv:2302.01806*.

Neng Wang, Hongyang Yang, and Christina Dan Wang. 2023. Fingpt: Instruction tuning benchmark for open-source large language models in financial datasets. *NeurIPS Workshop on Instruction Tuning and Instruction Following*.

Shaowen Wang, Linxi Yu, and Jian Li. 2024a. Lora-ga: Low-rank adaptation with gradient approximation. *arXiv preprint arXiv:2407.05000*.

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

Yaming Yang, Dilxat Muhtar, Yelong Shen, Yuefeng Zhan, Jianfeng Liu, Yujing Wang, Hao Sun, Denvy Deng, Feng Sun, Qi Zhang, et al. 2024a. Mtl-lora: Low-rank adaptation for multi-task learning. *arXiv preprint arXiv:2410.09437*.

Yibo Yang, Xiaojie Li, Zhongzhu Zhou, Shuaiwen Leon Song, Jianlong Wu, Liqiang Nie, and Bernard Ghanem. 2024b. Corda: Context-oriented decomposition adaptation of large language models. *arXiv preprint arXiv:2406.05223*.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. 2022. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12104–12113.

Fangzhao Zhang and Mert Pilanci. 2024. Spectral adapter: Fine-tuning in spectral space. *arXiv preprint arXiv:2405.13952*.

Wenyi Zhao, Rama Chellappa, P Jonathon Phillips, and Azriel Rosenfeld. 2003. Face recognition: A literature survey. *ACM computing surveys (CSUR)*, 35(4):399–458.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Yiyun Zhou, Wenkang Han, and Jingyuan Chen. 2025a. Revisiting applicable and comprehensive knowledge tracing in large-scale data. *arXiv preprint arXiv:2501.14256*.

Yiyun Zhou, Zheqi Lv, Shengyu Zhang, and Jingyuan Chen. 2024. Cuff-KT: Tackling learners' real-time learning pattern adjustment via tuning-free knowledge state-guided model updating.

Yiyun Zhou, Zheqi Lv, Shengyu Zhang, and Jingyuan Chen. 2025b. Disentangled knowledge tracing for alleviating cognitive bias. In *Proceedings of the ACM on Web Conference 2025*, pages 2633–2645.

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

Yuqi Zhu, Jia Li, Ge Li, YunFei Zhao, Zhi Jin, and Hong Mei. 2024b. Hot or cold? adaptive temperature sampling for code generation with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 437–445.

A Datasets

In our experiment, a total of 8 datasets are used for fine-tuning, and their descriptions are as follows.

- **databricks-dolly-15k:** databricks-dolly-15k is an open-source dataset with 15,000 high-quality, human-generated prompt/response pairs, created by over 5,000 Databricks employees in March and April 2023. It is designed for instruction tuning LLMs and includes expressive and diverse examples across various tasks such as brainstorming, content generation, classification, summarization, information extraction, closed QA, and open QA. Inspired by the behavioral categories in the InstructGPT (Ouyang et al., 2022), this dataset supports a wide range of instruction-following applications.

- **Lawyer-Instruct:** Lawyer-Instruct is an English-based conversational dataset derived from the original LawyerChat dataset. It features legal dialogue scenarios restructured into a format with clear instructions, inputs, and expected outputs. This redesigned format makes it particularly suitable for training supervised dialogue models.

- **US-Terms:** LegalLAMA is a comprehensive benchmark suite consisting of 8 sub-tasks, designed to evaluate the extent of legal knowledge acquired by PLMs during pre-training. US-Terms is one of these sub-tasks.

- **GenMedGPT-5k, clinic-10k:** GenMedGPT-5k and icliniq-10k both originate from ChatDoctor. GenMedGPT-5k contains a dataset of over 5,000 GPT-generated doctor-patient dialogues, while icliniq-10k includes a dataset of over 10,000 patient-doctor dialogues.

- **GSM8k:** GSM8K (Grade School Math 8K) is a dataset of 8.5K high-quality, linguistically diverse grade school math word problems. The dataset is designed to support the task of question answering on basic mathematical problems that require multi-step reasoning. We use the option-filled prompt shown in Appendix D to convert it into multiple-choice questions.

- **fingpt-fineval:** fingpt-fineval comes from FinGPT and includes Chinese multiple-choice questions instructions. However, the Llama model family does not support the Chinese language, so we used Google Cloud Translation[2] to translate it into English.

- **OpenOrca:** OpenOrca is an instruction-tuning dataset, with 2.91M samples derived from augmented FLAN (Longpre et al., 2023). It includes around 1M GPT-4 completions and 3.2M GPT-3.5 completions, organized as per ORCA's distribution (Mukherjee et al., 2023), aimed at training and evaluation in natural language processing tasks.

It is worth noting that we do not fine-tune the entire aforementioned datasets, but instead randomly select 1,000 samples with a random seed of 42, based on the following considerations: `(i)` **In real-world scenarios, fine-tuning samples are scarce (Van, 2023; Schäfer et al., 2024; Zhou et al., 2024; Pecher et al., 2024).** Data collection and labeling are costly, and high-quality samples are even harder to obtain for specific tasks. The scarcity of samples can easily lead to model overfitting, which poses a significant challenge for all PEFT methods. `(ii)` **Limited resources.** All resources are utilized on two NVIDIA A800-SXM4 (80G) GPUs, but most research institutions do not even have such configurations. We provide additional experiments with larger sample sizes in Appendix E to demonstrate the robustness of the proposed CoLA.

In addition, the descriptions of the MMLU and BBH datasets used in our experiments are as follows.

- **MMLU:** MMLU (Massive Multitask Language Understanding) is a benchmark designed to evaluate models in zero-shot and few-shot settings, covering 57 subjects across diverse fields like STEM, humanities, and social sciences. It tests both world knowledge and problem-solving ability, ranging from basic to advanced levels, and helps identify models' blind spots. The test challenges models to demonstrate extensive knowledge across multiple domains.

- **BBH:** BBH (BIG-Bench Hard) is a challenging subset of the BIG-Bench (bench authors, 2023), developed by Google and Stanford. It

[2] `https://cloud.google.com/translation-hub`

consists of 23 tasks that require multi-step reasoning, testing large language models' logical and reasoning abilities.

It's noted that we select tasks related to law (international law, jurisprudence, professional law) and medicine (anatomy, clinical knowledge, college medicine, human aging, human sexuality, medical genetics, professional medicine, virology) in MMLU for evaluation in the fields of law and medicine. We use the complete benchmark data for evaluation to ensure the adequacy and comprehensiveness of the experiments.

B Baselines

In our experiment, a total of 10 different PEFT methods are used to optimize the recent Llama models, described as follows.

- **Prompt Tuning:** Prompt Tuning introduces task-specific prompts to the input, updating only the prompt parameters while keeping the pretrained model's parameters frozen. It treats all tasks as generation tasks, with prompts being the focus of adaptation.
- **P-Tuning:** P-Tuning introduces trainable prompt embeddings optimized by a prompt encoder, eliminating manual prompt design. It allows prompt tokens to be added anywhere in the input sequence and includes anchor tokens to enhance performance.
- **IA^3:** IA^3 improves efficiency by integrating learned vectors into transformer models, reducing trainable parameters while maintaining performance and minimizing inference latency. This PEFT method involves multiplying model activations by three learned vectors, offering a more efficient alternative to LoRA with fewer parameters to update.
- **LoRA:** LoRA is a low-rank decomposition technique that reduces trainable parameters, speeding up fine-tuning and reducing memory usage by inserting trainable low-rank parameters into the original model weights.
- **DoRA:** DoRA decomposes the pre-trained weights into two components—magnitude and direction—to facilitate fine-tuning, using LoRA specifically for directional updates, which effectively minimizes the number of trainable parameters.
- **PiSSA:** PiSSA improves upon LoRA by initializing the adapter with principal singular values and vectors, optimizing the key components while freezing the "noisy" ones. This method leads to faster convergence and better performance than LoRA.
- **HydraLoRA:** HydraLoRA is an asymmetric fine-tuning architecture that effectively identifies and adapts to intrinsic data components, such as sub-domains or diverse tasks. It allocates distinct B matrices for task-specific features, while a shared A matrix integrates global information, enabling efficient parameter utilization and enhanced performance.
- **MOELoRA:** MOELoRA combines the benefits of multi-task learning and parameter-efficient fine-tuning by using multiple experts, each consisting of a pair of low-rank matrices, keeping trainable parameters minimal. The expert modules enable MOELoRA to handle task differences effectively and mitigate the negative impact of data imbalance on performance.
- **MTL-LoRA:** MTL-LoRA enhances low-rank adaptation (LoRA) by adding task-specific parameters, improving multi-task learning. It enables LLMs to adapt to diverse tasks efficiently, using fewer trainable parameters while capturing shared knowledge across tasks in low-dimensional spaces.
- **MoLA:** MoLA for Transformer-based models allows each layer to use a variable number of LoRA experts, with more experts allocated to higher layers, enhancing model effectiveness while maintaining a fixed total number of experts.

The PEFT methods related to LoRA mentioned above are used to optimize all linear modules (down_proj, k_proj, v_proj, q_proj, up_proj, gate_proj, o_proj).

C Two Types of Inference Modes

Codes 1 and 2 represent the code for two different inference modes, respectively. As can be observed, Code 1 is more suitable for classification tasks (*e.g.*, multiple-choice questions), as it is based on the model's logits for inference. When selecting the final answer, the decision is made directly by choosing the option with the highest probability. This mode does not involve a text generation process, so its results are relatively stable and consistent. The evaluation accuracy of this mode is not affected by the randomness of the generation process. In contrast, Code 2 is based on generation tasks (*e.g.*, text generation, question answering, etc.). It generates the entire text to obtain the answer (extracted via regular expressions, with different studies even using different extraction methods), and the quality of the generated output is significantly influenced by sampling strategies (*e.g.*, temperature). A trade-off must be made between diversity and accuracy in the generation process. For reproducibility, we select the first mode recommended by LlamaFactory for all experiments.

```
@torch.inference_mode()
def batch_inference_logit(self,
    batch_input: Dict[str, "torch.Tensor
    "]) -> List[str]:
    # self.choice_inputs: Encoding of
        the options in the instruction
    logits = self.model(**batch_input).
        logits
    lengths = torch.sum(batch_input["
        attention_mask"], dim=-1)
    word_probs = torch.stack([logits[i,
        lengths[i] - 1] for i in range(
        len(lengths))], dim=0)
    choice_probs = torch.nn.functional.
        softmax(word_probs[:, self.
        choice_inputs], dim=-1).detach()
    return [chr(ord("A") + offset.item()
        ) for offset in torch.argmax(
        choice_probs, dim=-1)]
```

Code 1: Inference mode with logit.

```
@torch.inference_mode()
def batch_inference_text(self,
    batch_input: Dict[str, "torch.Tensor
    "]) -> List[str]:
    outputs = self.model.generate(
        input_ids=batch_input["input_ids
            "],
        attention_mask=batch_input["
            attention_mask"],
        max_length=self.eval_args.
            max_answer_length,  #
            Adjustable maximum generated
             length
        num_beams=1,
        early_stopping=True
    )
    return [self.tokenizer.decode(output
        , skip_special_tokens=True) for
        output in outputs]
```

Code 2: Inference mode with text.

D Experimental Details

Some fine-tuning datasets (GSM8K and subsets of BBH) have instruction formats that are not based on multiple-choice questions, so they need to be extended through answer options to adapt to classification tasks. We generate additional options using the following option-filling prompt.

• The Prompt for Option Filling

Now there is a question about {subject} and a correct option. Please fill in the other incorrect options based on the question's context. Note that you should add the incorrect options, not solve the question.

—

Question: Henry and 3 of his friends order 7 pizzas for lunch. Each pizza is cut into 8 slices. If Henry and his friends want to share the pizzas equally, how many slices can each of them have?
Correct Option: A. 14
Answer:
B. 56
C. 8
D. 18

Question: Farmer Brown has 20 animals on his farm, all either chickens or cows. They have a total of 70 legs, all together. How many of the animals are chickens?
Correct Option: C. 5
Answer:
A. 20
B. 15
D. 70

Question: {question}
Correct Option: {correct_option}
Answer:
{incorrect_options}

We use the zero-cost API GLM-4-Flash[3] to accomplish the above task. In addition, the following evaluation template in our experiments is used, as shown in Figure 6.

Note that we present our experimental details, as shown in Tables 6 and 7.

E Additional Experimental Results

To verify the robustness of CoLA when faced with larger sample sizes (sample size = 1000, 2000,

[3]https://bigmodel.cn/dev/activities/free/glm-4-flash

Method	Llama-3.2-3B	FFT	Prompt Tuning	P-Tuning	IA^3	LoRA$_{r=8}$	LoRA$_{r=16}$	LoRA$_{r=24}$	LoRA$_{r=32}$	DoRA	PiSSA	HydraLoRA	CoLA	CoLA$^{\intercal}$	CoLA†	CoLA‡
#A | #B	-	-	-	-	-	1 | 1	1 | 1	1 | 1	1 | 1	1 | 1	1 | 1	1 | 3	1 | 3	2 | 3	2 | 3	2 | 3
%Param	-	-	0.0008	0.0530	0.0089	0.3770	0.7511	1.1224	1.4910	0.3770	0.3770	0.8267	0.7581	0.9406	0.9406	0.9406
Generality	23.01	32.87	24.71	25.02	24.56	26.28	31.25	32.88	34.90	26.38	<u>37.33</u>	25.37	**45.36****	**46.99****	**26.53**	**43.15****
Law	24.73	<u>26.28</u>	23.09	25.69	24.94	24.90	25.07	24.73	24.56	24.96	24.56	24.79	**27.51**	**26.85**	**24.89**	**24.99**
Medicine	24.03	<u>32.29</u>	23.28	25.39	24.24	24.71	25.39	25.80	25.87	24.57	27.10	24.51	**39.86****	**37.41****	**24.91**	**33.08***
Math	24.87	<u>53.92</u>	25.17	25.09	24.79	45.87	50.04	50.11	52.16	45.56	53.50	44.73	**56.71***	**56.79***	**41.72**	**52.86**
Finance	26.42	<u>42.03</u>	24.53	25.66	26.78	35.09	36.60	39.25	39.62	34.34	40.38	33.21	**39.62**	**41.51**	**38.93**	**40.32**

Table 5: Comparison of 0-shot performance (%) of different fine-tuning methods based on Llama-3.2-3B across multiple single domains.

• **USER.**
The following are multiple choice questions (with answers) about `{subject}`. Please provide only the correct option (one uppercase letter).
`{instruction}`
Answer:
• **ASSISTANT.**
`{correct_option}`

Figure 6: Evaluation Template

Hyperparameter	Setting
Batch Size	8
Train Epochs	5.0
Validation Size	0.1
Learning Rate	5e-5
Cutoff Length	1024
Gradient Accumulation Steps	8
Random Seed	42,43,44,45,46
Scheduler Type	cosine
Precision	fp16
Evaluation Strategy	steps
Optimizer	Adamw
GPU	two NVIDIA A800-SXM4 (80G) GPUs

Table 6: Experimental hyperparameter settings

3000, 4000, 5000, All), we selected PiSSA$_{r=16}$ and PiSSA$_{r=24}$ as the baselines for CoLA ($\#A = 1, \#B = 3$) and CoLA$^{\intercal}$ ($\#A = 2, \#B = 3$) in the generality domain, as shown in Table 8. From the table, it can be observed that as the sample size increases, CoLA and CoLA$^{\intercal}$ consistently outperform PiSSA$r = 16$ and PiSSA$r = 24$, which confirms the robustness of CoLA with respect to more samples. We also notice that having too many samples (ALL) does not necessarily lead to optimal performance, which may be due to the model's parameter scale not keeping up with the scaling law.

Method	Hyperparameter	Setting
Prompt Tuning	prompt_tuning_init_text	Answer the following question as required.\n
P-Tuning	num_virtual_tokens encoder_hidden_size encoder_num_layers encoder_reparameterization_type	20 256 2 MLP
IA3	target_modules	default
LoRA	r target_modules lora_alpha	8,16,24,32,64 down_proj, k_proj, v_proj, q_proj, up_proj, gate_proj, o_proj r × 2
DoRA	r	8
PiSSA	r	8
HydraLoRA	#A | #B	1 | 3,1 | 14
MOELoRA	#A | #B	8 | 8
MTL-LoRA	#A | #B	1 | 14
MoLA	#A | #B Type	8 | 8 Inverted-Triangle (MoLA-∇)
CoLA	r #A | #B	8 1 | 3,2 | 3,1 | 14, 4 | 10

Table 7: Hyperparameter settings used in our experiments for different PEFT methods.

Sample Size	1000	2000	3000	4000	5000	ALL
PiSSA$_{r=16}$	55.95	57.28	59.60	61.86	60.82	57.37
PiSSA$_{r=24}$	55.62	52.02	60.88	61.00	60.85	57.29
CoLA	**58.04**	**59.48**	**62.49**	**62.89**	**62.14**	**59.53**
CoLA$^{\intercal}$	**58.21**	**58.84**	**61.54**	**63.28**	**63.33**	**60.56**

Table 8: Performance comparison of CoLA and PiSSA in the generity domain based on Llama-3.1-8B as the sample size increases.