MiLoRA: Harnessing Minor Singular Components for Parameter-Efficient LLM Finetuning

Anonymous ACL submission

Abstract

Efficient finetuning of large language models (LLMs) aims to adapt the LLMs with reduced computational and memory costs. Previous LoRA-based approaches initialize the low-rank matrices with Gaussian distribution and zero values while keeping the original weight matrices frozen. However, the trainable model parameters optimized in an unguided subspace might interfere with the well-learned subspace of the pretrained weight matrices. In this paper, we propose MiLoRA, a simple yet effective LLM finetuning approach that only updates the minor singular components of the weight matrix while keeping the principal singular components frozen. It is observed that the minor matrix corresponds to the noisy or long-tail information, while the principal matrix contains important knowledge. The MiLoRA initializes the low-rank matrices within a subspace that is orthogonal to the principal matrix, thus the pretrained knowledge is expected to be well preserved. During finetuning, MiLoRA makes the most use of the less-optimized subspace for learning the labeled dataset. Extensive experiments on commonsense reasoning, math reasoning, instruction following and visual instruction following benchmarks present the superior performance of our method.

1 Introduction

Large language models (Brown et al., 2020b; Ouyang et al., 2022; Touvron et al., 2023; Jiang et al., 2023, LLMs) have demonstrated superior performance on various tasks (Zheng et al., 2023), such as math reasoning (Wang et al., 2024a) and question answering (Ivison et al., 2023). These models are pretrained with the next token prediction task (Brown et al., 2020b) on large web-scale data, then finetuned with instruction data as well as human preference data (Ouyang et al., 2022; Yu et al., 2023; Cui et al., 2023) for different downstream tasks. Fully finetuning is commonly employed to unlock the complete potential of LLMs, however, optimizing all model parameters necessitates substantial and restrictive computational resources (Touvron et al., 2023; Jiang et al., 2023), which hampers the utilization of LLMs across diverse scenarios.

Parameter-efficient finetuning (Hu et al., 2021, 2023; Liu et al., 2022; Zhang et al., 2024; Liu et al., 2024b, PEFT) aims at reducing the computational and GPU memory cost for finetuning of pretrained models. Low-rank adaptation (Hu et al., 2021, LoRA) is one of the most widely used PEFT methods for LLM finetuning. It assumes the change of linear model weights to be low-rank (Li et al., 2018; Aghajanyan et al., 2021). For each selected weight matrix, it only updates two low-rank matrices while keeping the pretrained weight frozen. During inference, the low-rank matrices are merged into the pretrained linear weights, thus no additional computational or memory cost is introduced. Recently, researchers have explored different LoRA-based variants for efficient LLM finetuning (Zhang et al., 2023; Pan et al., 2024; Kopiczko et al., 2024; Liu et al., 2024b; Meng et al., 2024). However, most existing LoRA-based methods randomly initialize the low-rank matrices and optimize the trainable parameters in an unguided subspace. We suspect this strategy may override important pretrained features, thus degrading the performance of low-rank adaptation methods (Dou et al., 2024).

In this paper, we propose **Mi**nor singular component based **Lo**w **R**ank **A**daptation (MiLoRA) for efficient LLM finetuning. MiLoRA has a similar framework as LoRA but employs a different initialization schedule. Specifically, a weight matrix $\boldsymbol{W}$ is decomposed with the singular value decomposition (SVD) algorithm. Based on the magnitude of the singular values, we divide $\boldsymbol{W}$ into two components: the principal matrix $\boldsymbol{W}_p$ corresponding to large singular values and the minor matrix $\boldsymbol{W}_m$ corresponding to small singular values. We argue that the principal matrix captures the essence of the

pretrained knowledge, whereas the minor matrix is suboptimal with noisy or long-tail information. It is supported by previous works (Hajimolahoseini et al., 2021; Sharma et al., 2024) that principal low-rank approximation can achieve comparable or even better performance to full parameters.

Motivated by these observations, we freeze the principal matrix $\boldsymbol{W}_p$ and adapt the minor singular components during finetuning. The low-rank matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ in LoRA framework are initialized with the minor matrix $\boldsymbol{W}_m$. Since the trainable low-rank matrices are initialized in a subspace orthogonal to the principal matrix, MiLoRA is expected to effectively learn from finetuning tasks while better preserving and utilizing the pretrained knowledge. To maintain the capability of the pretrained model at the start of finetuning, vanilla LoRA explicitly initializes $\boldsymbol{B}$ with zeros. In contrast, MiLoRA naturally satisfies this requirement, as the pretrained weight matrix $\boldsymbol{W}$ equals the frozen principal part $\boldsymbol{W}_p$ plus the low-rank part $\boldsymbol{W}_m = \boldsymbol{B}\boldsymbol{A}$. We conduct extensive experiments on commonsense reasoning, math reasoning, instruction following and visual instruction following benchmarks. The experimental results show that MiLoRA consistently outperforms LoRA without sacrificing training or inference efficiency, such as commonsense reasoning (+1.6/+1.1 on LLaMA2-7B/LLaMA3-8B), math reasoning (+2.0 on LLaMA2-7B), instruction following (+1.4 on LLaMA2-7B) and visual instruction following (+1.4 on LLaVA1.5-7B).[1]

2 Preliminaries

Singular Value Decomposition Given a matrix $\boldsymbol{W} \in \mathbb{R}^{m\times n}$, its singular value decomposition is denoted as $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, where $\boldsymbol{U} = [u_1, u_2, \cdots, u_m] \in \mathbb{R}^{m\times m}$, $\boldsymbol{V} = [v_1, v_2, \cdots, v_n] \in \mathbb{R}^{n\times n}$. The columns of $\boldsymbol{U}$ are the left singular vectors, and the columns of $\boldsymbol{V}$ are the right singular vectors. The $\boldsymbol{\Sigma} \in \mathbb{R}^{m\times n}$ is a diagonal matrix containing the singular values of $\boldsymbol{W}$ in descending order. Without loss of generality, we suppose $m \leq n$ to simplify the notation. The SVD of $\boldsymbol{W}$ can be reformulated as

$$\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \sum_{i=1}^{m} \sigma_i u_i v_i^\top, \qquad (1)$$

where u_i and v_i are the ith column of $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively.

[1]Our code and model will be made public.

Low-Rank Adaptation The low-rank adaptation method (Hu et al., 2021, LoRA) assumes the updates of linear weight $\boldsymbol{W} \in \mathbb{R}^{m\times n}$ to be low-rank, thus models the changes with two trainable low-rank matrices $\boldsymbol{A} \in \mathbb{R}^{r\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{m\times r}$. The weight matrix can be decomposed as

$$\boldsymbol{W} = \boldsymbol{W}^{(0)} + \Delta\boldsymbol{W} = \boldsymbol{W}^{(0)} + \frac{\alpha}{r}\boldsymbol{B}\boldsymbol{A}, \qquad (2)$$

where $\boldsymbol{W}^{(0)}$ and $\Delta\boldsymbol{W}$ refer to the pretrained weight and weight change, respectively. The α and r are hyperparameters of scaling-factor and LoRA rank ($r \ll \min(m, n)$). During finetuning, the pretrained matrix $\boldsymbol{W}^{(0)}$ is kept frozen. It significantly diminishes the number of trainable parameters as both $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices are low-rank. The $\boldsymbol{B}$ matrix is initialized with zero while $\boldsymbol{A}$ matrix adopts a random Gaussian distribution with zero mean value. This initialization strategy ensures the $\Delta\boldsymbol{W} = 0$ at the beginning of training. The LoRA method only modifies the linear matrices in the Transformer model. The low-rank matrices can be easily merged into the pretrained linear matrix to get updated for inference, which does not require additional computing and GPU memory compared with full finetuning. However, the vanilla LoRA method fails to select the optimal subspace for updating the model parameters, as the low-rank matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are randomly initialized. This might potentially detract from the pretrained knowledge encoded in the pretrained weight matrix.

3 Methodology

The proposed MiLoRA is a simple yet effective PEFT approach that selects the minor singular components of the weight matrices for optimization. As observed in LASER (Sharma et al., 2024), the minor singular components of weight matrices contain noisy or long-tail information, while the principal singular components are responsible for important features across tasks. Therefore, MiLoRA is designed to effectively learn from the finetuning dataset by adapting minor singular components, while maintaining the knowledge encoded in the pretrained model. On the contrary, vanilla LoRA fails to constrain the optimization subspace as its low-rank matrices are randomly initialized. Specifically, suppose $m \leq n$, MiLoRA splits each linear weight matrix into two matrices according to the corresponding singular values: principal matrix

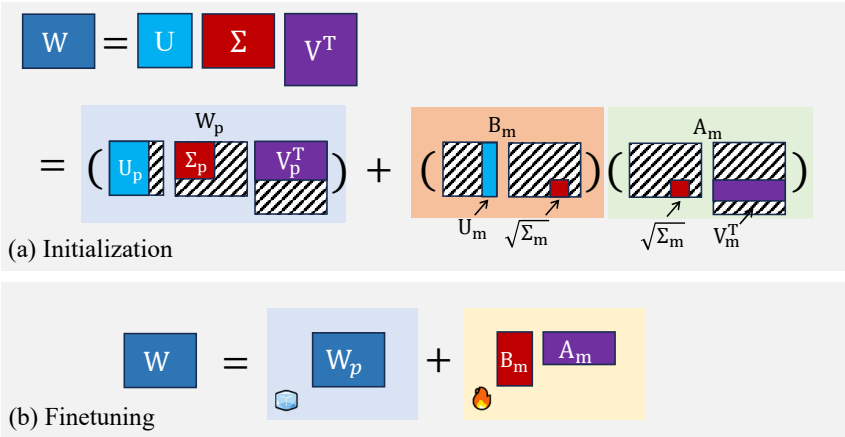

Figure 1: (a) MiLoRA method splits the pretrained linear weight matrix into two parts, the minor singular component $(\boldsymbol{U}_m, \boldsymbol{V}_m, \boldsymbol{\Sigma}_m)$ is used to initialize the low-rank matrices $\boldsymbol{A}_m$ and $\boldsymbol{B}_m$ in the LoRA framework. (b) During finetuning, only the low-rank matrices $\boldsymbol{A}_m$ and $\boldsymbol{B}_m$ are updated while the principal matrix $\boldsymbol{W}_p$ is frozen, which is similar to the practice of LoRA method.

$\boldsymbol{W}_p$ and minor matrix $\boldsymbol{W}_m$ (see Figure 1):

$$\begin{aligned} \boldsymbol{W} &= \boldsymbol{W}_p + \boldsymbol{W}_m \\ &= \sum_{i=1}^{m-r} \sigma_i u_i v_i^\top + \sum_{i=m-r+1}^{m} \sigma_i u_i v_i^\top, \end{aligned} \tag{3}$$

where the singular values are in descending order ($\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m \geq 0$), and the hyperparameter r is the number of minor singular values in the $\boldsymbol{W}_m$ matrix.

The decomposition can also be reformulated in a matrix form. The $\boldsymbol{U}$ matrix in the SVD of $\boldsymbol{W}$ can be reformulated as $\boldsymbol{U} = [\boldsymbol{U}_p, \boldsymbol{U}_m]$, where $\boldsymbol{U}_p = [u_1, u_2, \cdots, u_{m-r}]$ and $\boldsymbol{U}_m = [u_{m-r+1}, u_{m-r+2}, \cdots, u_m]$ are left singular vectors corresponding to principal and minor singular values, respectively. The $\boldsymbol{V}$ and $\boldsymbol{\Sigma}$ matrices can be reformulated similarly. The SVD of $\boldsymbol{W}$ can be expressed as:

$$\begin{aligned} \boldsymbol{W} &= \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \boldsymbol{U}_p\boldsymbol{\Sigma}_p\boldsymbol{V}_p^\top + \boldsymbol{U}_m\boldsymbol{\Sigma}_m\boldsymbol{V}_m^\top \\ &= \boldsymbol{W}_p + \boldsymbol{W}_m \end{aligned} \tag{4}$$

During the finetuning process, instead of freezing the entire pretrained weight matrix, we just keep the principal singular components $\boldsymbol{W}_p$ fixed to preserve the pretrained knowledge. The minor matrix $\boldsymbol{W}_m$ is used to initialize low-rank matrices $\boldsymbol{A}_m$ and $\boldsymbol{B}_m$ (see Figure 1):

$$\begin{aligned} \boldsymbol{W}_m &= \boldsymbol{U}_m\boldsymbol{\Sigma}_m\boldsymbol{V}_m^\top \\ &= (\boldsymbol{U}_m\sqrt{\boldsymbol{\Sigma}_m})(\sqrt{\boldsymbol{\Sigma}_m}\boldsymbol{V}_m^\top) = \boldsymbol{B}_m\boldsymbol{A}_m, \end{aligned} \tag{5}$$

This strategy has two benefits: 1) It encourages the model to learn in the less-optimized subspace spanned by the minor singular vectors, thus reducing the interference with the well-learned pretrained knowledge encoded in the principal singular components. 2) Unlike vanilla LoRA which requires tuning the scaling factor α and the initialization hyperparameters, ours does not require any hyperparameter except the rank r. One potential variant of MiLoRA is to impose strict constraints on the orthogonality between $\boldsymbol{W}_m$ and $\boldsymbol{W}_p$ during fine-tuning. However, we adopt a softer approach by guiding the optimization direction solely through initialization, resulting in a method that is simpler, more training-efficient, and more flexible. Our experiment results show that our scheme works quite well in balancing learning from the finetuning dataset and preserving pretrained knowledge.

4 Experiments

To evaluate MiLoRA against other PEFTs, we conduct experiments on both NLP and multimodal tasks covering 18 datasets. We benchmark against LoRA (Hu et al., 2021) and PiSSA (Meng et al., 2024). All of our experiments are conducted on 8 NVIDIA L40 GPUs unless otherwise specified.

- LoRA (Hu et al., 2021) reparameterizes the weight update $\Delta\boldsymbol{W}$ with two trainable low-rank matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, while freezing the pretrained weight $\boldsymbol{W}$. They use a random Gaussian initialization for A and zero for B, so $\Delta\boldsymbol{W} = \boldsymbol{B}\boldsymbol{A}$ is zero at the beginning of training.
- PiSSA (Meng et al., 2024) shares the same framework as LoRA, but employs a more sophisticated initialization approach. They initialize $\boldsymbol{A}$ and $\boldsymbol{B}$ with principal singular values and

Model	PEFT	BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg.
ChatGPT	–	73.1	85.4	68.5	78.5	66.1	89.8	79.9	74.8	77.0
LLaMA2-7B	LoRA†	**69.8**	79.9	79.5	83.6	**82.6**	79.8	64.7	**81.0**	77.6
	PiSSA	67.6	78.1	78.4	76.6	78.0	75.8	60.2	75.6	73.8
	MiLoRA	67.6	**83.8**	**80.1**	**88.2**	82.0	**82.8**	**68.8**	80.6	**79.2**
LLaMA3-8B	LoRA†	**70.8**	85.2	**79.9**	91.7	84.3	84.2	71.2	79.0	80.8
	PiSSA	67.1	81.1	77.2	83.6	78.9	77.7	63.2	74.6	75.4
	MiLoRA	68.8	**86.7**	77.2	**92.9**	**85.6**	**86.8**	**75.5**	**81.8**	**81.9**

Table 1: Commonsense reasoning evaluation results for LLaMA2-7B and LLaMA3-8B on eight tasks. †Results are cited from Liu et al. (2024b). Bold numbers indicate the highest performance scores for each dataset across the different PEFT methods for the corresponding model.

Method	GSM8K	MATH	Avg.
Full FT†	66.5	19.8	43.2
LoRA	60.6	16.9	38.7
PiSSA	58.2	15.8	37.0
MiLoRA	**63.5**	**17.8**	**40.7**

Table 2: Math reasoning evaluation results for LLaMA2-7B on GSM8K and MATH. †Results are cited from Yu et al. (2023).

Method	Win-rate
GPT-4-0613†	91.2
GPT-3.5-turbo-0613†	91.8
LLaMA2-Chat-7B†	87.3
LLaMA2-7B + LoRA	85.9
LLaMA2-7B + PiSSA	85.6
LLaMA2-7B + MiLoRA	**87.3**

Table 3: Instruction tuning evaluation results for instruction-tuned LLaMA2 7B with Alpaca-Eval v1.0. †Results are cited from Ivison et al. (2023).

singular vectors of the pre-trained weight $\boldsymbol{W}$. Given that the principal components capture the essence of a matrix, PiSSA is *expected* to better approximate full finetuning by changing the essential parts while freezing the "noisy" parts.

4.1 Experiments on Large Language Model

Models and Datasets We compare MiLoRA with baselines on three different types of LLM downstream tasks.

• **Commonsense reasoning:** We finetune LLaMA2-7B (Touvron et al., 2023) and LLaMA3-7B (AI@Meta, 2024) on Commonsense170K (Hu et al., 2023). Eight commonsense reasoning datasets are used for evaluation, including 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-e, ARC-c (Clark et al., 2018), and OBQA (Mihaylov et al., 2018). The task is formulated as a multiple-choice problem. We report accuracy (%) for all datasets on the best checkpoint chosen by the validation set loss.

• **Math reasoning:** We finetune LLaMA2-7B (Touvron et al., 2023) on the MetaMathQA dataset (Yu et al., 2023), which contains 395K samples augmented from the training set of GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021). We use test sets of GSM8K and MATH for evaluation and report results on the last checkpoint. For each test set, we report the exact match ratio against the ground truth.

• **Instruction following:** We finetune LLaMA2-7B with Ultrafeedback (Cui et al., 2023) following previous works (Wu et al., 2024a,b). We use Alpaca-Eval v1.0 (Li et al., 2023a) for evaluation and report win-rate against text-davinci-003 (Brown et al., 2020a) using GPT4-0613 (OpenAI, 2024) as the annotator.

Implementation Details We use the same hyperparameter configurations as Hu et al. (2023) without tuning for all methods. Details can be found in Appendix A.2. We denote this hyperparameter setup as LLM-Adapters. By default, we use a rank of 32. As math reasoning has a large training dataset, we set the rank to 64.

Results We report results in Table 1, Table 2 and 3 for commonsense reasoning, math reasoning and instruction following, respectively. For commonsense reasoning, we also include the ChatGPT baseline reported in Liu et al. (2024b) as a reference, which is obtained with gpt-3.5-turbo API using a zero-shot Chain of Thought (Wei et al., 2022). For math reasoning, we add the full finetuning results from Yu et al. (2023) as a reference. For instruction following, we include the results of GPT-4-0613,

Method	VQAv2	GQA	VisWiz	SQA	VQAT	POPE	MMBench	Avg.
Full FT†	78.5	61.9	50.0	66.8	58.2	85.9	64.3	66.5
LoRA†	79.1	**62.9**	47.8	68.4	58.2	86.4	66.1	66.9
PiSSA	77.5	60.6	39.2	67.3	54.3	87.0	61.4	63.9
MiLoRA	**79.2**	62.1	**53.3**	**70.6**	**58.7**	**87.9**	**66.1**	**68.3**

Table 4: Visual instruction tuning evaluation results for LLaVA1.5-7B on seven vision-language tasks. †We directly use checkpoints from Liu et al. (2024a) to reproduce the results of baseline methods.

GPT-3.5-turbo-0613, and LLaMA2-Chat-7B from Ivison et al. (2023) as a reference.

As can be seen, MiLoRA consistently surpasses all baseline methods across different datasets and LLMs, indicating that MiLoRA is a highly effective finetuning method. Specifically, for commonsense reasoning, MiLoRA outperforms LoRA and PiSSA by an average of 1.6 and 5.4 accuracy points on LLaMA2-7B, and by 1.1 and 6.5 points on LLaMA3-8B. For math reasoning, MiLoRA improves over LoRA and PiSSA by 2.0 and 3.7 averaged EM points on LLaMA2-7B, respectively. However, it still underperforms full finetuning by an average of 2.5 EM score. This suggests that the PEFT methods still have room for improvement to fully match the performance of full finetuning. For instruction following, MiLoRA achieves the best results, surpassing LoRA and PiSSA by 1.4 and 2.7 in win-rate scores, respectively.

4.2 Experiments on Vision-Language Model

Models and Datasets To further examine if MiLoRA can remain competitive on multimodal finetuning tasks, we further conduct visual instruction tuning tasks on LLaVA1.5-7B (Liu et al., 2024a), which consists of an LLM, Vicuna-1.5-7B (Peng et al., 2023) and a vision encoder, CLIP ViT-L/336px (Radford et al., 2021). The visual instruction tuning is performed in a multitask setting, including VQA (Hudson and Manning, 2019; Marino et al., 2019; Schwenk et al., 2022), OCR (Mishra et al., 2019; Sidorov et al., 2020), region-level VQA (Krishna et al., 2017; Mao et al., 2016), visual conversation (Liu et al., 2024a), and language conversation data. The finetuned models are evaluated on seven different multimodal benchmarks: VQA-v2 (Goyal et al., 2017), GQA (Hudson and Manning, 2019), VizWiz (Gurari et al., 2018), SQA (Lu et al., 2024), VQAT (Singh et al., 2019), POPE (Li et al., 2023b), and MMBench (Liu et al., 2024c).

Implementation Details We follow the settings of Liu et al. (2024a) to construct the training data and prompt template. For the hyperparameter configuration, we follow the LoRA configuration provided by Liu et al. (2024a) without tuning for all PEFT methods for a fair comparison. Details can be found in Appendix A.4.

Results Table 4 shows the experiment results. As can be seen, MiLoRA achieves the best results, outperforming LoRA and PiSSA by 1.4 and 4.4 average accuracy points, respectively. We also note that in this setup, fully fine-tuning is less effective than LoRA. As a result, methods like PiSSA, which aim to closely approximate fully fine-tuning, may lose their advantage. In contrast, MiLoRA improves performance by retaining more of the pre-trained knowledge, thereby circumventing this issue.

5 Understanding MiLoRA

In this section, we conduct experiments to further understand MiLoRA. By default, we use the fine-tuned LLaMA2-7B model on math reasoning from our main experiments for analyses.

5.1 Comparison with More PEFTs

To further demonstrate the effectiveness of MiLoRA, we compare MiLoRA with additional popular baselines, such as rsLoRA (Kalajdzievski, 2023), LoRA+ (Hayou et al., 2024), DoRA (Liu et al., 2024b), AdaLoRA (Zhang et al., 2023), LoRA-GA (Wang et al., 2024b). We follow the math and code experiment settings in Wang et al. (2024b). Specifically, we finetune LLaMA2-7B with MiLoRA on MetaMathQA 100K and Code-Feedback 100K for one epoch using rank 8 and compare MiLoRA with baseline results reported in Wang et al. (2024b). Table 5 shows the results. We report the training cost of all methods relative to LoRA, which requires 3.58 hours of training on the math task using a single NVIDIA-L40 GPU. As can be seen, MiLoRA consistently outperforms all baselines, improving the best PEFT baseline LoRA-GA by 1.1 and 4.6 points on GSM8K and Human-eval, respectively. The training efficiency

Method	GSM8K	Human-eval	Training Cost
rsLoRA	45.6	16.0	$\times$ 1.00
LoRA+	52.1	18.2	$\times$ 1.00
DoRA	53.1	19.8	$\times$ 1.58
AdaLoRA	50.7	17.8	$\times$ 1.07
LoRA-GA	53.6	19.8	$\times$ 1.00
MiLoRA	**54.7**	**24.4**	$\times$ 1.00

Table 5: Comparison with more baselines on math and code tasks. Baseline GSM8K and Human-eval results are cited from Wang et al. (2024b).

Method	principal	Random	Minor
GSM8K	60.7	63.2	**64.0**
MATH	14.6	15.5	**16.1**

Table 6: Performance of MiLoRA when initializing with principal, random sampled, and minor singular components.

of MiLoRA is also very high. The time for SVD decomposition in MiLoRA is less than six minutes and negligible when compared with the total training time. Therefore, the training time required of MiLoRA is consistent with that of LoRA, while DoRA and AdaLoRA require additional training time overhead.

5.2 Is Minor Singular Component Adaptation Important?

To investigate the influence of singular components of varying magnitudes on finetuning performance, we conduct experiments on math reasoning tasks using the LLaMA2-7B model. Specifically, we finetune LLaMA2-7B on the MetaMathQA 395K dataset for 1 epoch with a maximum sequence length of 512 to save computation. We initialize the low-rank matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ with principal, minor, and randomly sampled singular components, and report the evaluation results in Table 6. As can be seen, initializing with the minor singular components achieves the best performance across both datasets. This underscores the importance of adapting the minor singular components during the finetuning process.

5.3 How does the Matrix Update ΔW Compare to W?

We investigate the relationship between $\Delta\boldsymbol{W}$ and $\boldsymbol{W}$ of different methods, where $\Delta\boldsymbol{W}$ is the difference between the finetuned and pre-trained weights. Following the analysis method in Hu et al. (2021), we project $\boldsymbol{W}$ and $\Delta\boldsymbol{W}$ onto the r-dimensional subspace of $\Delta\boldsymbol{W}$ by computing $\boldsymbol{U}^T\boldsymbol{W}\boldsymbol{V}$ and $\boldsymbol{U}^T\Delta\boldsymbol{W}\boldsymbol{V}$, with $\boldsymbol{U}/\boldsymbol{V}$ being the left/right top r singular-vector matrix of $\Delta\boldsymbol{W}$. As a comparison, we also compute $\boldsymbol{U}^T\boldsymbol{W}\boldsymbol{V}$ by replacing $\boldsymbol{U}, \boldsymbol{V}$ with the top r singular vectors of $\boldsymbol{W}$ or a random matrix. The results are shown in Table 7. Following Hu et al. (2021), we use the query weight in the middle layer (16th) of the model for analysis and use the Frobenius norm as the measurement of magnitude.

We make two key observations. First, $\Delta\boldsymbol{W}$ of LoRA and MiLoRA have a similar correlation with $\boldsymbol{W}$, both slightly stronger than a random matrix, indicating that they amplify directions not emphasized in $\boldsymbol{W}$. In contrast, $\Delta\boldsymbol{W}$ of PiSSA shows a strong correlation with $\boldsymbol{W}$. This is attributed to PiSSA's direct optimization of the principal singular components. Second, the amplification factor varies significantly across different methods. Compared to LoRA, we find the amplification factor of MiLoRA is significantly smaller. This indicates that MiLoRA has a reduced impact on the features which already present in $\boldsymbol{W}$. See Table 13 of Appendix B for analyzing the weight matrix of MLP down projection in the same layer, which demonstrates similar trends.

To further analyze the characteristics of $\Delta\boldsymbol{W}$ for different methods, we measure the normalized subspace similarity (Hu et al., 2021) between the subspaces spanned by top r singular vectors of $\Delta\boldsymbol{W}$ and the subspace spanned by top r singular vectors, bottom r singular vectors, and a random r singular vectors of $\boldsymbol{W}$. We define the normalized similarity metric based on the Grassmann distance following Hu et al. (2021) (See Appendix A.1 for more details) and report the results in Figure 2. From the left subfigure, it is clear that PiSSA learns in directions closely aligned with the top singular vectors of the pretrained weights, while LoRA and MiLoRA do not significantly optimize in these directions. Specifically, MiLoRA focuses on optimizing directions associated with the bottom singular vectors of the pretrained weights. The right subfigure further shows that weight matrices in both MLP and self attention modules of MiLoRA learn in directions more closely aligned with the bottom singular vectors, especially the MLP modules.

5.4 MiLoRA Forgets Less than Baselines

A hypothesis explanation for why MiLoRA outperforms baselines is that pre-trained knowledge is more retained. To test this hypothesis, we follow Kalajdzievski (2024) and use cross-entropy as

$\|\boldsymbol{W}\|_F = 97.36$	$\boldsymbol{W}$	Random	LoRA	PiSSA	MiLoRA
$\|\boldsymbol{U}^\top \boldsymbol{W} \boldsymbol{V}\|_F =$	43.64	1.50	1.82	17.57	1.86
$\|\boldsymbol{U}^\top \Delta\boldsymbol{W} \boldsymbol{V}\|_F =$	-	-	68.18	55.79	44.95
Amplification factor	-	-	37.46	3.18	24.17

Table 7: The Frobenius norm of $\boldsymbol{U}^\top \boldsymbol{W} \boldsymbol{V}$ where $\boldsymbol{U}$ and $\boldsymbol{V}$ are the left/right top r singular vectors of either $\boldsymbol{W}$, a random matrix, or $\Delta\boldsymbol{W}$ of PEFTs. The weight matrices are taken from the query projection of the 16th layer of LLaMA2-7B.

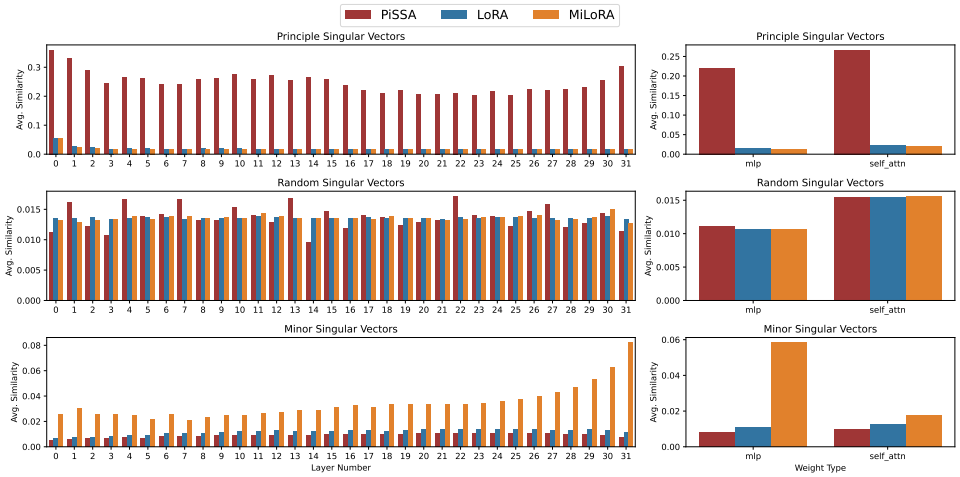

Figure 2: **Left:** The subspace similarity averaged by all modules in a layer. **Right:** The subspace similarity averaged by all layers for a specific module type.

Method	LoRA	PiSSA	MiLoRA
Forgetting loss	3.24	6.07	2.54

Table 8: Forgetting loss of various PEFT methods.

the metric for measuring forgetting. This is the usual next token prediction loss used when training LLMs, except that the target next token is replaced by the distribution predicted by the pre-trained base model. We evaluate the forgetting metric on the LLaMA2-7B model finetuned in math reasoning using the WikiText-103 test dataset following Kalajdzievski (2024). As shown in Table 8, MiLoRA exhibits the lowest forgetting loss, which is consistent with our hypothesis that MiLoRA makes the least modification to pre-trained knowledge. In contrast, the loss of PiSSA is significantly higher than that of the other methods.

5.5 Comparison Between MiLoRA and PiSSA

Concurrent with our research, Meng et al. (2024) have recently introduced a low-rank adaptation method called PiSSA. PiSSA shares a similar framework as MiLoRA, but adapts the principal singular components. We argue that PiSSA and MiLoRA are fundamentally different.

Hyp. Setup	PEFT	GSM8K	MATH	Avg.
PiSSA	LoRA	41.5	5.8	23.6
	PiSSA	**51.3**	**7.6**	**29.4**
	MiLoRA	40.0	5.2	22.6
LLM-Adapters	LoRA	56.6	10.8	33.7
	PiSSA	51.3	10.4	30.8
	MiLoRA	**58.6**	**11.6**	**35.1**

Table 9: Math reasoning evaluation results for LLaMA2-7B with PiSSA hyperparameters and our (LLM-Adapters) hyperparameters.

Motivation: PiSSA is designed to approximate full finetuning by adapting the principal singular components, which are believed to capture the essence of the weight matrices. In contrast, our method MiLoRA aims to adapt to new tasks **while maximally retaining the base model's knowledge**. To achieve this, we instead finetune the minor singular components of the weight matrices, which are less important for the pretrained weight matrices.

Performance: The PiSSA paper claims that PiSSA outperforms both LoRA and adapting minor singular components. However, our investigation suggests that this claim is likely due to the specific

hyperparameters used in their experiments. To verify this, we replicate their experimental setup using the same MetaMathQA 100K dataset and compare the performance of PiSSA, LoRA, and MiLoRA under both PiSSA's hyperparameters and our own. Specifically, we choose a rank of 128 where PiSSA obtains the best performance in their paper and finetunes for 1 epoch for all setups following PiSSA paper. The results are summarized in Table 9.

We have **several observations**. **First**, under the PiSSA hyperparameter setup, PiSSA does indeed outperform both LoRA and MiLoRA, consistent with the claims made in the original PiSSA paper. **Second**, we find that all methods perform better in our hyperparameter setup, suggesting that our hyperparameter configuration is more effective. PiSSA configuration of hyperparameters has a combination of a small learning rate (2e-5) and a large batch size (128). This could potentially result in slow learning speed and sub-optimal performance for common PEFTs, as the optimal learning rate for common PEFTs is generally much higher than that for full finetuning (Lialin et al., 2023; Biderman et al., 2024). **Third**, in our hyperparameter setup, MiLoRA performs the best, outperforming LoRA and PiSSA by 1.4 and 4.3 average scores, respectively. In summary, when using an appropriate hyperparameter configuration, our proposed MiLoRA method achieves markedly superior performance compared to PiSSA.

6 Related Work

Parameter-Efficient Finetuning of LLMs. After pretraining, LLMs are finetuned with instruction data or human preference data to adapt to different downstream tasks (Ouyang et al., 2022; Yu et al., 2023; Cui et al., 2023). Parameter-efficient finetuning (Lialin et al., 2023, PEFT) explores effective approaches to reduce the computational resources required during finetuning while managing to match the performance of full finetuning. Previous PEFT methods can be grouped into three lines: adapter-, LoRA- and prompt-based methods. Adapter-based methods insert additional trainable feedforward submodules into each Transformer layer (Houlsby et al., 2019; Pfeiffer et al., 2021a,b). However, they introduce additional computational costs during inference. LoRA-based methods (Hu et al., 2021; Liu et al., 2024b; Zhang et al., 2023) model the changes of selected linear weights as low-rank matrices. During finetuning, given a linear matrix, only two low-rank matrices are optimized while the pretrained weight matrix is frozen. Prompt-based methods (Lester et al., 2021; Li and Liang, 2021; Liu et al., 2022) add additional soft prompts to the input tokens. During training, only soft prompts are updated while the pretrained model parameters are fixed. Among these PEFT methods, the LoRA-based approaches are widely used for LLM fine-tuning because they are easy to implement and do not introduce computational overhead during inference.

LoRA and Its Variants LoRA (Hu et al., 2021) reparameterizes the weight update with two trainable low-rank matrices, while freezing the pretrained weights. With this lightweight decomposition, LoRA reduces storage and task-switching overhead by sharing the pretrained models across multiple tasks. Since then, researchers have explored and proposed different LoRA variants for PEFT. AdaLoRA (Zhang et al., 2023) and ALoRA (Liu et al., 2024d), adaptively determine the rank of LoRA module in each weight matrix according to the importance score. The rsLoRA (Kalajdzievski, 2023) modifies the LoRA with the appropriate scaling factor to improve the performance of large ranks. The DoRA (Liu et al., 2024b) method decomposes the pretrained weight into the magnitude and directional components, then finetunes both for better performance. Concurrent with our work, PISSA (Meng et al., 2024) and LoRA-GA (Wang et al., 2024b) propose to better approximate full finetuning by only updating the principal singular components or aligning the gradients of low-rank updates to that of full finetuning.

7 Conclusion

In this paper, we introduce MiLoRA, a simple yet effective low-rank adaption method for LLMs. MiLoRA effectively learns on finetuning tasks while better preserving the pretrained knowledge by adapting the minor singular components of pretrained weight matrices. We investigate the effectiveness of MiLoRA on a wide range of LLM evaluation benchmarks, including commonsense reasoning, math reasoning and instruction-following, and vision-language model evaluation benchmarks. Experiment results demonstrate that MiLoRA consistently outperforms LoRA and PiSSA without sacrificing training or inference efficiency. We hope that our work will inspire future research on parameter-efficient finetuning of LLMs.

Limitations

Due to limited computational resources, we only examine the effectiveness of MiLoRA on the LLaMA-family LLMs and tasks like commonsense reasoning, math reasoning and instruction following benchmarks following previous work (Liu et al., 2024b; Wu et al., 2024b). We examine and compare MiLoRA with baselines using the hyperparameter configurations provided in the previous works (Hu et al., 2023; Liu et al., 2024b) instead of exhaustive hyperparameter search for each task. We leave the exploration of MiLoRA on other tasks and other LLMs like Mistral and Gemma as future work.

References

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7319–7328, Online. Association for Computational Linguistics.

AI@Meta. 2024. Llama 3 model card.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John Patrick Cunningham. 2024. LoRA learns less and forgets less. *Transactions on Machine Learning Research*. Featured Certification.

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020a. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, and et.al. 2020b. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

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. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota.

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, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2023. Ultrafeedback: Boosting language models with high-quality feedback. *arXiv preprint arXiv:2310.01377*.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Loramoe: Alleviate world knowledge forgetting in large language models via moe-style plugin. *Preprint*, arXiv:2312.09979.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913.

Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. 2018. Vizwiz grand challenge: Answering visual questions from blind people. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3608–3617.

Habib Hajimolahoseini, Mehdi Rezagholizadeh, Vahid Partovinia, Marzieh S. Tahaei, Omar Mohamed Awad, and Yang Liu. 2021. Compressing pre-trained language models using progressive low rank decomposition. In *NeurIPS*.

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

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

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 *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 2790–2799.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

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, Singapore.

Drew A Hudson and Christopher D Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702*.

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

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

Damjan Kalajdzievski. 2024. Scaling laws for forgetting when fine-tuning large language models. *arXiv preprint arXiv:2401.05605*.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M Asano. 2024. VeRA: Vector-based random matrix adaptation. In *The Twelfth International Conference on Learning Representations*.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *Int. J. Comput. Vision*, 123(1):32–73.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic.

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

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of ACL*, pages 4582–4597, Online.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023a. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. 2023b. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 292–305, Singapore.

Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. 2023. Scaling down to scale up: A guide to parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.15647*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024a. Visual instruction tuning. *Advances in neural information processing systems*, 36.

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, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 61–68.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. 2024c. Mmbench: Is your multi-modal model an all-around player? In *ECCV*.

Zequan Liu, Jiawen Lin, Wei Zhu, and Xing Tian. 2024d. Alora: Allocating low-rank adaptation for fine-tuning large language models. In *Proceedings of NAACL*.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2024. Learn to explain: multimodal reasoning via thought chains for science question answering. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11–20.

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. 2019. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

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*.

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*.

Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. 2019. Ocr-vqa: Visual question answering by reading text in images. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 947–952.

OpenAI. 2024. [link].

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, and et.al. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744.

Rui Pan, Xiang Liu, Shizhe Diao, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. 2024. LISA: Layerwise importance sampling for memory-efficient large language model fine-tuning. *arXiv preprint arXiv:2403.17919*.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021a. 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, Online.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2021b. UNKs everywhere: Adapting multilingual language models to new scripts. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10186–10203, Online and Punta Cana, Dominican Republic.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*.

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 Le Bras, and Yejin Choi. 2019. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China. Association for Computational Linguistics.

Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. 2022. A-okvqa: A benchmark for visual question answering using world knowledge. In *ECCV*, page 146–162, Berlin, Heidelberg.

Pratyusha Sharma, Jordan T. Ash, and Dipendra Misra. 2024. The truth is in there: Improving reasoning in language models with layer-selective rank reduction. In *The Twelfth International Conference on Learning Representations*.

Oleksii Sidorov, Ronghang Hu, Marcus Rohrbach, and Amanpreet Singh. 2020. Textcaps: A dataset for image captioning with reading comprehension. In *Computer Vision – ECCV 2020*, pages 742–758, Cham.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. 2019. Towards vqa models that can read. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8309–8318.

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

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. 2024a. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*.

Shaowen Wang, Linxi Yu, and Jian Li. 2024b. Lora-ga: Low-rank adaptation with gradient approximation. Technical report. ArXiv 2407.05000.

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

Muling Wu, Wenhao Liu, Xiaohua Wang, Tianlong Li, Changze Lv, Zixuan Ling, Jianhao Zhu, Cenyuan Zhang, Xiaoqing Zheng, and Xuanjing Huang.

2024a. Advancing parameter efficiency in fine-tuning via representation editing. *arXiv preprint arXiv:2402.15179*.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. 2024b. Reft: Representation finetuning for language models. *arXiv preprint arXiv:2404.03592*.

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*.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*.

Renrui Zhang, Jiaming Han, Chris Liu, Aojun Zhou, Pan Lu, Yu Qiao, Hongsheng Li, and Peng Gao. 2024. LLaMA-adapter: Efficient fine-tuning of large language models with zero-initialized attention. In *The Twelfth International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

A Detailed Experiment Setups

A.1 Similarity Metric Between Subspaces

Following Hu et al. (2021), we use the measure $\phi(\boldsymbol{A}, \boldsymbol{B}) = \frac{\|\boldsymbol{A}^\top \boldsymbol{B}\|_F^2}{r}$ to measure the similarity between two column orthonormal matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{d\times r}$, where F donotes the Frobenius norm. The value of $\phi(\boldsymbol{A}, \boldsymbol{B})$ ranges from 0 to 1, where 1 indicates complete overlap and 0 indicates complete separation.

Given two matrices $\Delta\boldsymbol{W}$ and $\boldsymbol{W}$, we extract r left singular vectors from each to form the subspace matrices for them, which are denoted as $\Delta\boldsymbol{W}_s$ and $\boldsymbol{W}_s$, and then the subspace similarity is computed using $\phi(\Delta\boldsymbol{W}_s, \boldsymbol{W}_s)$.

A.2 Our Hyperparameter Setup for LLM

Table 10 shows our detailed hyperparameters. This setup follows LLM-adapters (Hu et al., 2023), therefore we denote it as LLM-adapters setup.

Hyperparameters	ComR	MathR	InsF
Rank r	32	64	32
α of LoRA	64	128	64
α of PiSSA/MiLoRA	32	64	32
Dropout	0.05		
Optimizer	AdamW		
LR	3e-4		
LR Scheduler	Linear		
Batch size	16		
Warmup Steps	100		
Epochs	3		
Placement	query, key, value, MLP up, MLP down		

Table 10: Our hyperparameter configuration on the commonsense reasoning (ComR), math reasoning (MathR) and instruction-following (InsF) tasks.

A.3 PiSSA Hyperparameter Setup for LLM

Table 12 shows the detailed hyperparameters from the PiSSA paper, which we denoted as PiSSA hyperparameter setup.

A.4 More Implementation Details

We use the implementation of LLM-Adapters (Hu et al., 2023)[2] for commonsense reasoning, the implementation of PiSSA (Meng et al., 2024)[3] for math reasoning and the implementation of open-instruct[4] for instruction tuning. For hyperparameters of finetuning the vision-language model, we follow the LoRA configuration provided by Liu et al. (2024a) without tuning for all PEFT methods for a fair comparison. Table 11 shows the details.

B More Experiment Results

We present additional results about the relationship between $\Delta\boldsymbol{W}$ and $\boldsymbol{W}$ in Table 13, in which the weight matrices are taken from the MLP down projection of the 16th layer of LLaMA2-7B.

C Broader Impacts

The MiLoRA method enhances model performance with lower training costs. Teams or users with limited computational resources can finetune large models using MiLoRA, promoting the broader application of large models across diverse groups.

[2] https://github.com/AGI-Edgerunners/LLM-Adapters

[3] https://github.com/GraphPKU/PiSSA

[4] https://github.com/allenai/open-instruct

Visual instruction tuning hyperparameters	
Rank r	128
α of LoRA	256
α of PiSSA/MiLoRA	128
Dropout	0.05
Optimizer	AdamW
LR	2e-4
LR Scheduler	cosine
Batch Size	16
Warmup Ratio	0.03
Epochs	1
Placement	query, key, value, output, gate, MLP up, MLP down

Table 11: Our hyperparameter configurations applied in visual instruction tuning tasks.

PiSSA hyperparameters	
α	Same as rank r
Dropout	0.0
Optimizer	AdamW
LR	2e-5
LR Scheduler	cosine
Batch Size	128
Warmup Ratio	0.03
Epochs	1
Placement	query, key, value, output, gate, MLP up, MLP down

Table 12: The detailed hyperparameter configurations used by PiSSA(Meng et al., 2024).

However, the PEFT methods could potentially be exploited to finetune models for malicious purposes. It is essential to investigate rules and strategies to prevent the misuse of PEFT methods.

$\lVert \boldsymbol{W} \rVert_F = 123.0$	$\boldsymbol{W}$	Random	LoRA	PiSSA	MiLoRA
$\lVert \boldsymbol{U}^\top \boldsymbol{W} \boldsymbol{V} \rVert_F =$	33.34	1.17	1.32	6.98	1.29
$\lVert \boldsymbol{U}^\top \Delta \boldsymbol{W} \boldsymbol{V} \rVert_F =$	-	-	77.02	74.34	56.61
Amplification factor	-	-	58.35	10.65	43.88

Table 13: The Frobenius norm of $\boldsymbol{U}^\top \boldsymbol{W} \boldsymbol{V}$ where $\boldsymbol{U}$ and $\boldsymbol{V}$ are the left/right top r singular vectors of either $\boldsymbol{W}$, a random matrix, or $\Delta \boldsymbol{W}$ of PEFTs. The weight matrices are taken from the MLP down projection of the 16th layer of LLaMA2-7B.