SuLoRA: Subspace Low-Rank Adaptation for Parameter-Efficient Fine-Tuning

Chenhao Ding[1], **Jiangyang Li**[2], **Songlin Dong**[2] *, **Xinyuan Gao**[1], **Yuhang He** [2], **Yihong Gong**[2]

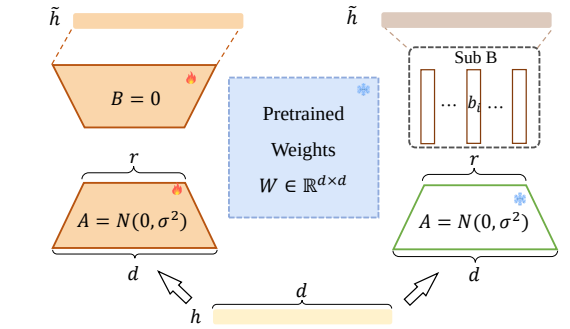
[1]School of Software Engineering, Xi'an Jiaotong University

[2]State Key Laboratory of Human-Machine Hybrid Augmented Intelligence, National Engineering Research Center for Visual Information and Applications, and Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University

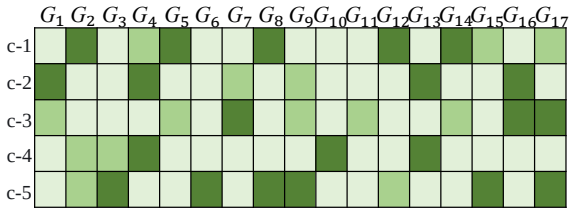
{dch225739, dsl972731417}@stu.xjtu.edu.cn, heyuhang@xjtu.edu.cn, ygong@mail.xjtu.edu.cn

Abstract

As the scale of large language models (LLMs) grows and natural language tasks become increasingly diverse, Parameter-Efficient Fine-Tuning (PEFT) has become the standard paradigm for fine-tuning LLMs. Among PEFT methods, LoRA is widely adopted for not introducing additional inference overhead. However, existing LoRA's shared parameter space paradigm introduces parameter interference, leading to a gap in generalization performance for specific tasks compared to full fine-tuning. To address this issue, we propose a parameter-separated low-rank adapter, called Subspace Low-Rank Adaptation (SuLoRA). The core idea of SuLoRA is to account for task differences by decomposing LoRA's parameter matrix into multiple independent subspaces and assigning them differentially to distinct tasks. This prevents interference across tasks and enhances the effectiveness of low-rank adaptation. Additionally, SuLoRA achieves higher rank expansion by freezing the A matrix, further improving generalization capability. We conduct extensive experiments on various NLP tasks, demonstrating that SuLoRA significantly outperforms LoRA in trainable parameter efficiency and overall model performance. Furthermore, we validate SuLoRA's effectiveness in domain generalization and multi-modal tasks, showcasing its strong generalization ability.



(a) Comparison between LoRA (left) and our method SuLoRA (right).



(b) Activation levels of the parameter space for different classes.

Figure 1: (a) Comparison between LoRA (left) and our method SuLoRA (right). The core of our method is to partition the internal parameter space of LoRA and select different parameter spaces for different tasks, avoiding interference between tasks. (b) Activation of the internal parameter space in LoRA B. G and c denote the parameter subspace and task, respectively. Darker colors indicate higher levels of activation. Taking STS-B as an example, we find that the activation of the internal parameter space in LoRA varies across different tasks.

1 Introduction

Large language models (LLMs), such as GPT (Radford, 2018), Llama (Touvron et al., 2023a), and DeepSeek (Liu et al., 2024), have emerged as the standard in natural language processing (NLP) owing to their exceptional comprehension capabilities (Brown et al., 2020). However, as the scale of these models and the complexity of tasks continue to grow, full fine-tuning (FT) has become impractical due to its substantial computational cost. In response, parameter-efficient fine-tuning (PEFT) methods (Ding et al., 2023), including low-rank adaptation (LoRA)(Hu et al., 2021), adapters(He et al., 2021), and prompt-based approaches (Lester et al., 2021a) have been successfully applied across a diverse range of NLP tasks.

Among these methods, LoRA (Hu et al., 2021) is widely employed because it does not incur additional inference overhead. As illustrated in Fig. 1(a) (left), LoRA approximates the update of pre-trained weights (W) by decomposing them into a low-rank matrix (A) and a high-rank matrix (B).

* Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2025, pages 5334–5349
July 27 - August 1, 2025

By applying low-rank decomposition to the incremental updates of the pre-trained weights, LoRA significantly reduces the memory overhead associated with fine-tuning. However, compared to full fine-tuning, low-rank approximations may result in a considerable generalization performance gap (Ren et al., 2024). Consequently, enhancing the performance of low-rank approximations while minimizing computational overhead has emerged as a critical challenge.

Recently, AsyLoRA (Zhu et al., 2024) proposed a method that requires no additional parameters by leveraging the inherent asymmetry of LoRA matrices. Specifically, the A matrix captures fundamental features, while the B matrix utilizes these features to generate task-specific outputs. Consequently, fine-tuning B is inherently more effective than fine-tuning A. This method enhances the effective rank by freezing the A matrix during fine-tuning. However, despite successfully expanding the rank, it yields minimal performance improvement. This motivates a deeper investigation into the functional role of the B matrix in LoRA.

In Fig. 1(b), we partition the parameter space of the LoRA B matrix and visualize the activations of these parameter groups (G) across different classes. It can be observed that distinct classes exhibit significant differences in their parameter preferences; for instance, the parameter groups predominantly activated by 'c-1' and 'c-2' are nearly mutually exclusive. However, traditional LoRA does not account for such inter-task preference differences, as it employs a shared parameter space paradigm. This approach may lead to interference among the parameter spaces of different classes during fine-tuning (Liu et al., 2023; Ghiassian et al., 2020), ultimately affecting the performance of LoRA.

> *How can we effectively distinguish the parameter preferences across tasks to mitigate such interference and enhance the model's generalization capability?*

Given our objective to segregate the parameter space during fine-tuning to accommodate the distinct preferences of different tasks (*e.g.*, varying categories or datasets) and mitigate inter-task interference, we propose a low-rank adapter that partitions the parameter space, termed **Su**bspace **L**ow-**R**ank **A**daptation (**SuLoRA**).

As shown in Fig. 1 (a), we first freeze matrix A, which is insensitive to fine-tuning performance, to improve its effective rank and thereby expand the trainable parameter space of matrix B (Zhu et al., 2024). In contrast to traditional LoRA, which shares a single low-rank adaptation matrix across all task instances, SuLoRA partitions the parameter space of matrix B into N subspaces (N > r). During training, SuLoRA dynamically selects r subspaces based on the parameter preferences of each task, constructing a dedicated matrix B for each task to mitigate inter-task interference. Experimental results demonstrate that this strategy yields significant performance improvements.

To validate the effectiveness and generalization capability of SuLoRA, we conducted extensive experiments across various NLP tasks and extended our evaluation to domain generalization and multimodal tasks. Experimental results demonstrate that SuLoRA consistently outperforms LoRA while utilizing fewer parameters and a lower rank, and it even surpasses full-parameter fine-tuning across multiple tasks. In summary, our method makes the following main contributions:

- Through experiments and analysis, we found that different tasks exhibit significant differences in parameter preferences during the fine-tuning of LoRA, which plays a crucial role in enhancing the performance of low-rank approximations.
- We propose a parameter-separated low-rank adapter, SuLoRA, which allocates distinct subparameter spaces for different tasks, effectively mitigating inter-task interference.
- Extensive experiments conducted across three domain tasks demonstrate the effectiveness of SuLoRA, which outperforms LoRA and SOTA methods in both parameter efficiency and performance.

2 Related Work

Low-Rank Adaptation (LoRA). LoRA (Hu et al., 2021) is a lightweight fine-tuning method designed to reduce the cost of adapting large pre-trained models for specific tasks. By introducing low-rank matrices to approximate weight updates, LoRA avoids full parameter fine-tuning, significantly reducing computational and memory overhead. Variants and extensions of LoRA have been widely applied in natural language processing (Zhu et al., 2024; Zhang et al., 2023; Ren et al., 2024; Tian et al., 2024) and computer vision (Yao et al., 2024; Shen et al., 2024), for fine-tuning models like BERT (Alaparthi and Mishra, 2020) and GPT (Radford, 2018).

Notably, AsyLoRA (Zhu et al., 2024) has revealed an asymmetry between the roles of the A and B matrices in LoRA. The study demonstrates that the performance of a randomly initialized A matrix is comparable to that of a fine-tuned one. Consequently, a strategy was proposed to extend the rank by freezing the A matrix and exclusively fine-tuning the B matrix. However, this work did not delve deeply into the parameter space of the B matrix, resulting in limited performance improvements. Furthermore, MELoRA (Ren et al., 2024) achieves efficient inference by integrating multiple sub-LoRAs. However, these sub-LoRAs are shared across different tasks, which is fundamentally distinct from our approach.

Parameter Efficiency in Existing Methods. Adapter methods (Zhang et al., 2023; Fu et al., 2022; Gao et al., 2024) enhance fine-tuning efficiency by inserting small, trainable modules at each layer of a pre-trained model. In contrast, prompt tuning (Lester et al., 2021b; Liu et al., 2021) and prefix tuning (Li and Liang, 2021) adjust the model's output by adding trainable prompts or prefixes to the input, without modifying the internal weights. Unlike these methods, LoRA replaces full model weight updates with low-rank matrices, avoiding large-scale modifications to the pre-trained model parameters. Moreover, as this work focuses on LoRA improvements and a fair comparison with the above approaches is challenging, these methods were excluded from our experiments.

Parameter Interference. Parameter conflict (Bengio et al., 2020) refers to the interference between model parameters, leading to performance degradation. In reinforcement learning, parameter conflict typically manifests as incompatibility between states, where learning in one state may overwrite knowledge in another, resulting in a decline in model performance and generalization ability (Liu et al., 2023; Ghiassian et al., 2020).

This paper examines the performance degradation caused by parameter sharing in LoRA fine-tuning from the perspective of parameter conflicts and addresses it by allocating distinct sub-parameter spaces to each task.

3 Method

3.1 Preliminaries

LoRA (Low-Rank Adaptation) is a parameter-efficient fine-tuning (PEFT) method that integrates trainable low-rank decomposition matrices into each layer of Transformer models, enabling fine-tuning without the need for a full update of the model's original weights. As shown in Fig. 1(a) (left), the computation of a linear layer can be represented as:

$$\tilde{h} = Wh, \tag{1}$$

where $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ is the pre-trained weight matrix, with d_{in} and d_{out} representing the input and output dimensions, respectively.

LoRA adjusts the weights by introducing low-rank decomposition matrices, achieving parameter-efficient adaptation. The computation is as follows:

$$\tilde{h} = Wh + \Delta Wh = Wh + \alpha \cdot BAh, \tag{2}$$

where $\Delta W = BA$ is the trainable adjustment matrix composed of low-rank matrices $A \in \mathbb{R}^{r \times d_{\text{in}}}$ and $B \in \mathbb{R}^{d_{\text{out}} \times r}$, where the rank r satisfies $r \ll \min(d_{\text{in}}, d_{\text{out}})$. The hyperparameter $\alpha > 0$ controls the influence of the adjustment matrix on the overall weights.

During fine-tuning, LoRA updates only the low-rank decomposition matrices (LoRA A and B), leaving the original pre-trained weights unchanged. This enables efficient task-specific adaptation with reduced computational overhead by training independent LoRA modules for each downstream task.

3.2 Parameter Activation Sensitivity

This work explores parameter preferences across tasks in natural language tasks and examines their activations. For task i, the normalized gradient update magnitude is used to represent the update behavior of the parameter groups:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \frac{\|W_i \odot \nabla W_i\|_F}{\|W_i\|_F + \epsilon}, \tag{3}$$

where N is the total number of parameters in the parameter groups involved in the computation, ϵ is a small constant, ∇W_i is the gradient of W_i, and $\| \cdot \|_F$ represents the Frobenius norm of the parameters, which normalizes the parameter scale. The term $\|W_i \odot \nabla W_i\|_F$ performs an element-wise multiplication between the gradient and the parameters, which measures directional consistency. If W_i and ∇W_i are aligned in the same direction, the update is more effective.

Leveraging the normalized gradient obtained from Eq. 3, we can measure the activation level of parameters relative to themselves, avoiding the influence between parameters of different scales, and

by comparison, determine which parameter groups are effectively activated.

Therefore, a parameter group is defined as effectively activated when $\mathcal{L} > 0.7$ (indicated by dark green in Fig. 1(b)). If $\mathcal{L} < 0.3$, the parameter group is considered dormant (indicated by light green in Fig. 1(b)). Values in between are classified as an intermediate state (indicated by grass green in Fig. 1(b)).

We conducted experiments on the fine-tuned RoBERTa-base (Liu, 2019) model using LoRA with a rank of 4 and calculated the parameter activation of LoRA B from five types of sentence pairs with different similarity levels in STS-B (Cer et al., 2017). Fig. 1(b) shows the parameter activation scores of LoRA B in the fifth transformer (Vaswani, 2017) layer.

Our study reveals that parameter activation in LoRA B varies across different tasks. For instance, in STS-B, the activated parameter groups in class 0 exhibit significant differences in parameter preferences compared to those in class 1. The conventional parameter-sharing mechanism in LoRA inevitably overlooks these preferences, potentially leading to interference between task-specific parameter learning and, consequently, affecting the effectiveness of model fine-tuning.

To address this issue, we decompose the LoRA parameter matrix into multiple independent subspaces and assign them separately to different tasks to capture each task's preferences. Meanwhile, fine-tuning the B matrix is inherently more effective for specific tasks than fine-tuning the A matrix (Zhu et al., 2024). Consequently, we choose to freeze the A matrix to enhance the model's generalization capability and improve the scalability of the rank.

3.3 Subspace Low-Rank Adaptation

Through analysis of the prior section, we propose Subspace Low-Rank Adaptation (SuLoRA). As shown in Fig. 2, it divides LoRA B matrix into a set of N ($N > r$) distinct parameter subspaces $\{b_i\}$. By dynamically selecting r subspaces from $\{b_i\}$, SuLoRA customizes the LoRA B matrix for each input instance. The low-rank adaptation expression of SuLoRA is as follows:

$$\begin{aligned}\tilde{h} &= Wh + \Delta Wh \\ &= Wh + \alpha \cdot (\text{concat}_{i=0}^{r} b_i)\, Ah \\ &= Wh + \alpha \cdot BAh,\end{aligned} \tag{4}$$

where $b_i \in \mathbb{R}^{d_{\text{out}} \times 1}$ represents a selected parameter subspace, and $A \in \mathbb{R}^{r \times d_{\text{in}}}$.

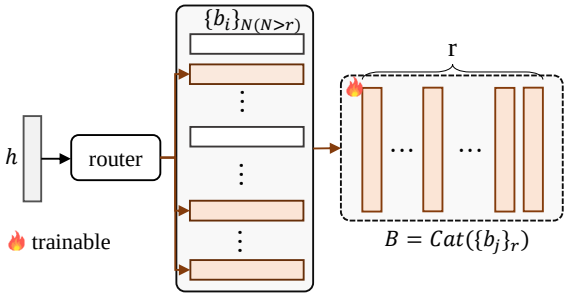


Figure 2: Parameter subspace selection of LoRA B matrix in SuLoRA. SuLoRA treats different parameter subspaces as experts and allocates r parameter subspaces to each instance through a routing method based on hidden features.

3.4 Parameter Subspace Selection

To allocate a dedicated parameter space for each instance, we adopt an instance-based routing method, inspired by the mixture of experts (MoE) (Shazeer et al., 2017) approach, to assign r parameter subspaces for each instance based on its hidden features. Compared to traditional token-based routing methods, this approach is significantly more memory-efficient.

Specifically, the router first computes the average over the sequence dimension of the input hidden features $h \in \mathbb{R}^{\text{seq} \times d_{\text{in}}}$, resulting in the averaged hidden features $\bar{h} = \text{Avg}(h)$.

Next, we allocate $\bar{h}$ using a set of learnable weights $W_q \in \mathbb{R}^{N \times d_{\text{in}}}$. Combined with top-k selection, this allows us to identify r parameter subspaces tailored to the specific instance:

$$g_B = \text{top}_r\left(\frac{\exp(W_q \bar{h})}{\sum_i \exp(W_q \bar{h}_i)}\right), \tag{5}$$

where g_B represents the selected parameter group consisting of the chosen $\{b_i\}$.

Through careful parameter selection, we can effectively mitigate parameter interference between tasks based on the parameter preferences of different tasks, creating a distinct parameter space for each instance and achieving parameter separation to address the interference issue in LoRA.

4 Experimental Setup

4.1 Datasets

NLP tasks. Based on previous work (Ren et al., 2024), we adopt two widely-used benchmarks: GLUE (Wang, 2018) and INSTRUCTEVAL (Chia et al., 2023). - The **GLUE** benchmark covers various natural language understanding (NLU) tasks,

Method	#Params	CoLA	SST-2	MRPC	QNLI	STS-B	RTE	MNLI	QQP	Avg.
FT	184M	69.21	95.64	89.22	93.78	91.59	82.49	89.98	92.05	87.82
AdaLoRA	1.27M	70.86	95.95	90.22	94.28	91.39	87.36	90.27	92.13	88.83
MELoRA	1.33M	70.67	96.41	90.98	94.39	91.83	87.94	90.35	91.77	89.29
AsyLora$_{r=8}$	0.67M	67.46	95.49	88.38	93.71	91.02	77.24	90.36	91.56	86.90
AsyLora$_{r=16}$	1.33M	69.73	95.64	89.46	93.74	91.18	80.14	90.08	91.98	87.74
LoRA$_{r=8}$	1.33M	69.73	95.57	89.71	93.76	91.86	85.32	90.47	91.95	88.38
SuLoRA$_{r=4}$	**0.34M**	71.22	96.55	91.02	94.12	91.84	87.01	90.06	91.51	89.17
SuLoRA$_{r=8}$	0.67M	**71.86**	**96.79**	**92.07**	**94.75**	**92.36**	**88.39**	**90.52**	**92.20**	**89.87**

Table 1: **Different adaptation methods on the GLUE benchmark for natural language understanding tasks.** We report the overall (matched and mismatched) accuracy for MNLI, Matthew's correlation for CoLA, Pearson correlation for STS-B, and accuracy for other tasks. Higher is better for all metrics. #Params represents the number of trainable parameters in each adaptation method. **Boldface** highlights the best results according to the corresponding metrics, while the second-best results are underlined.

including classification, semantic similarity, paraphrase detection, and natural language inference. It contains datasets such as CoLA (Xia et al., 2024), SST-2 (Socher et al., 2013), MRPC (Dolan and Brockett, 2005), QNLI (Rajpurkar et al., 2016), STS-B (Cer et al., 2017), RTE (Bentivogli et al., 2009b; Bar-Haim et al., 2014; Giampiccolo et al., 2007; Bentivogli et al., 2009a), MNLI (Williams et al., 2018), and QQP. - **INSTRUCTEVAL** focuses on instruction-following tasks and includes MMLU (Hendrycks et al., 2020), DROP (Dua et al., 2019), HumanEval (HEval) (Chen et al., 2021), and BBH (Srivastava et al., 2023). We fine-tune Llama2-7b (Touvron et al., 2023b) on cleaned Alpaca data and evaluate performance based on INSTRUCTEVAL.

Visual domain generalization tasks use the DomainBed benchmark, which contains VLCS (Fang et al., 2013), PACS (Li et al., 2017), and Office-Home (Venkateswara et al., 2017) datasets. The original 80% training and 20% testing splits are followed. This benchmark assesses image classification domain generalization across different style environments.

Multi-modal tasks. Following (Shen et al., 2024), we conduct instruction tuning on a subset of Vision-Flan (Xu et al., 2023), which consists of 187 tasks with up to 1,000 instances per task, totaling 182,167 examples. To further verify generalization, we also test on Text-VQA (Singh et al., 2019), VSR (Liu et al., 2022), CIFAR-100 (Krizhevsky et al., 2009), and MNIST (LeCun, 1998) datasets.

4.2 Comparison Methods

We conduct a comprehensive and fair comparison of our proposed method, SuLoRA, with conventional full fine-tuning (FT) approaches, the classical LoRA method (Hu et al., 2021), and other LoRA-based improvements, such as AdaLoRA (Zhang et al., 2023), MELoRA (Ren et al., 2024), and AsyLoRA (Zhu et al., 2024), across multiple tasks. Notably, similar to AsyLoRA, our approach freezes the A matrix in LoRA during training, focusing solely on training the B matrix, whereas LoRA, AdaLoRA, and MELoRA concurrently train both the A and B matrices. For further details, please refer to Appendix A.2.

4.3 Implementation Details

In all experiments, we only fine-tune the projection matrices W_Q and W_V (Zhang et al., 2023). Experiments are conducted on NVIDIA A800 GPUs, and results are averaged over 3 random seeds.

- For the **GLUE benchmark**, we use RoBERTa-base as the backbone model. To ensure fair comparison, the rank of LoRA is set to 8. When the rank $r = 4$, the number of parameter subspaces is set to $N = 8$; when $r = 8$, $N = 16$. Since the number of trainable parameters in SuLoRA is independent of N when the rank is fixed, we explore N from {6, 8, 10, 12} and report the best performance.
- For the **INSTRUCTEVAL benchmark**, the backbone is LLaMA-2-7B, with training on the Alpaca dataset and 2k randomly selected samples as the development set. Following INSTRUCTEVAL (Chia et al., 2023) settings, we apply 5-shot direct prompting for MMLU, 3-shot for BBH and DROP (dev), and 0-shot for HEval. We use AdamW optimizer, train for 3 epochs to match baseline epochs, apply a linear learning rate scheduler starting from 3×10^{-4}, and use a batch size of 16. The number of parameter subspaces is set to twice the rank.

Method	VLCS		PACS		OfficeHome	
	ID	OOD	ID	OOD	ID	OOD
LoRA $_{r=8}$	$73.51_{\pm 0.62}$	$56.43_{\pm 1.96}$	$94.94_{\pm 0.56}$	$\mathbf{75.58_{\pm 0.92}}$	$78.54_{\pm 1.49}$	$74.46_{\pm 0.40}$
LP	$75.58_{\pm 1.66}$	$71.70_{\pm 1.04}$	$81.62_{\pm 0.34}$	$61.73_{\pm 1.25}$	$58.38_{\pm 0.76}$	$68.59_{\pm 0.22}$
Full Fine-tuning	$76.21_{\pm 1.95}$	$64.87_{\pm 6.44}$	$\mathbf{98.15_{\pm 0.56}}$	$74.90_{\pm 2.43}$	$\mathbf{80.67_{\pm 1.22}}$	$63.23_{\pm 0.64}$
AdaLoRA$_{r=8}$	$76.85_{\pm 2.40}$	$64.92_{\pm 1.75}$	$91.98_{\pm 2.80}$	$71.89_{\pm 1.10}$	$77.12_{\pm 0.95}$	$74.30_{\pm 0.35}$
MELoRA$_{r=8}$	$77.72_{\pm 2.22}$	$71.01_{\pm 1.63}$	$92.59_{\pm 2.63}$	$71.45_{\pm 1.03}$	$77.62_{\pm 0.89}$	$75.88_{\pm 0.28}$
AsyLoRA$_{r=8}$	$77.40_{\pm 2.30}$	$\underline{75.81_{\pm 1.65}}$	$92.45_{\pm 2.68}$	$72.55_{\pm 1.03}$	$77.66_{\pm 0.89}$	$77.72_{\pm 0.32}$
AsyLoRA$_{r=16}$	$\underline{79.10_{\pm 1.41}}$	$75.40_{\pm 1.24}$	$93.52_{\pm 0.20}$	$73.76_{\pm 0.67}$	$77.63_{\pm 0.84}$	$\underline{77.85_{\pm 0.33}}$
SuLoRA$_{r=8}$	$\mathbf{79.60_{\pm 1.25}}$	$\mathbf{76.80_{\pm 1.40}}$	$\underline{96.77_{\pm 0.15}}$	$\underline{75.48_{\pm 0.50}}$	$\underline{78.64_{\pm 0.78}}$	$\mathbf{79.23_{\pm 0.30}}$

Table 2: **DomainBed results**. ID and OOD denote in-domain and out-of-domain test error, respectively. For OOD we report the average performance across different environments.

Method	#Params	MMLU	DROP	HEval	BBH
Llama2	-	45.96	31.55	12.20	32.04
FT	7B	**47.30**	29.12	12.80	32.72
LoRA	33.6M	45.64	32.46	15.09	32.40
QLoRA	33.6M	45.40	28.97	15.24	32.81
AdaLoRA	33.6M	45.96	31.94	14.02	32.85
MELoRA	33.6M	46.50	32.74	<u>16.19</u>	<u>32.98</u>
AsyLoRA	16.8M	46.35	<u>32.58</u>	15.97	32.83
SuLoRA	16.8M	<u>46.68</u>	**32.77**	**16.29**	**33.27**

Table 3: **Results on INSTRUCTEVAL for instruction-following tasks.** We report the exact match for MMLU, DROP and BBH, pass@1 for HumanEval. Higher is better for all metrics. The best results are in **bold** and the second-best results are <u>underlined</u>.

Method	Text-VQA	VSR	CIFAR-100	MNIST	Avg
LLaVA$_{\text{Align}}$	32.62	50.16	58.04	52.79	48.40
FT	37.26	53.76	**63.73**	**94.27**	**62.26**
LoRA $_{r=8}$	39.20	53.27	46.88	82.95	55.58
AdaLoRA	40.59	53.73	49.31	83.16	56.70
MELoRA	<u>40.72</u>	53.65	57.24	85.19	59.20
AsyLoRA	40.63	<u>52.81</u>	57.29	85.32	59.01
SuLoRA	**42.31**	**53.92**	<u>60.22</u>	<u>86.57</u>	<u>60.76</u>

Table 4: **Zero-shot for Multi-modal Evaluation.** We report the performance of our method on the Text-VQA, VSR, CIFAR-100, and MNIST. LLaVA$_{\text{Align}}$ indicates the stage-one LLaVA-v1 with only feature alignment but not visual instruction tuning. The best results are in **bold** and the second-best results are <u>underlined</u>.

- For the **DomainBed benchmark**, we adopt an ImageNet-pretrained ViT model for image classification and domain generalization. We fine-tune ViT on different environments (LabelMe, Cartoon, Clipart) from VLCS, PACS, and Office-Home datasets respectively.
- For the **Multi-modal tasks**, Llava1-13B is used as the backbone. Experiments on Vision-Flan subset, Text-VQA, VSR, CIFAR-100, and MNIST further demonstrate the model's diverse capabilities.

5 Results

5.1 Results on NLP tasks

Results on GLUE. Table 1 presents a comparison of our method with other approaches on the GLUE benchmark using the same backbone. We can observe that, under the same LoRA rank setting, our method outperforms LoRA on all GLUE datasets, achieving an Avg improvement of **1.49**. Even with a rank of 4, SuLoRA achieves better performance than LoRA on 5 out of 8 datasets (with an Avg improvement of **0.79**). Additionally, compared to MELoRA, when both methods have the same number of parameters as LoRA, SuLoRA shows superior performance at rank 8, with an Avg improvement of **0.58**.

It is worth noting that SuLoRA demonstrates more significant improvements on datasets with limited training data, such as CoLA, MRPC, and RTE. We believe this is because SuLoRA avoids overfitting on limited data by separating the parameter spaces. Additionally, by freezing the LoRA A matrix, SuLoRA enhances generalization ability. The performance of SuLoRA on the remaining datasets further demonstrates that it is stable and effective for different NLU tasks.

Results on INSTRUCTEVAL. To further validate the effectiveness of our method, we conducted tests on the INSTRUCTEVAL benchmark dataset. As shown in Table 3, under the same backbone, SuLoRA demonstrated superior performance compared to other methods, achieving the best results on all datasets except for MMLU. Compared to LoRA, we achieved improvements of **1.04**, **0.31**, **1.20**, and **0.87** on MMLU, DROP, HEval, and BBH, respectively. The performance on the INSTRUCTEVAL benchmark further demonstrates

Method	#Params	CoLA	SST-2	MRPC	QNLI	STS-B	RTE	MNLI	QQP	Avg.
$\hat{\mathbf{B}}_{\text{cat}}\hat{\mathbf{A}}$	0.67M	71.21	95.76	90.20	94.27	91.78	87.12	**90.30**	**91.86**	89.06
$\hat{\mathbf{B}}\hat{\mathbf{A}}_{\text{cat}}$	0.67M	70.88	96.44	90.44	94.17	91.71	**87.36**	90.26	91.07	89.04
$B\hat{\mathbf{A}}_{\text{cat}}$	0.34M	71.09	95.68	90.36	**94.33**	91.57	85.06	90.29	91.22	88.70
$\hat{\mathbf{B}}_{\text{cat}}A$ (Ours)	0.34M	**71.22**	**96.55**	**91.02**	94.12	**91.84**	87.01	90.06	91.51	**89.17**

Table 5: **Comparison of Different Design Strategies for SuLoRA.** The experiments are conducted on the GLUE benchmark. $\hat{(\cdot)}$ indicates that the parameter is trainable, and "cat" refers to the use of a parameter space decomposition strategy. In SuLoRA, A is frozen, while B uses the parameter space decomposition strategy. **Boldface** indicates the best result, while the second-best result is underlined.

Method	MMLU	DROP	HEval	BBH
Llama2	45.96	31.55	12.20	32.04
$\hat{\mathbf{B}}_{\text{cat}}\hat{\mathbf{A}}$	**46.71**	32.49	16.08	33.11
$\hat{\mathbf{B}}\hat{\mathbf{A}}_{\text{cat}}$	45.88	31.75	15.75	32.81
$B\hat{\mathbf{A}}_{\text{cat}}$	45.74	32.15	15.53	32.70
$\hat{\mathbf{B}}_{\text{cat}}A$ (Ours)	46.68	**32.77**	**16.29**	**33.27**

Table 6: **Comparison of Different Design Strategies for SuLoRA.** The experiments are conducted on the INSTRUCTEVAL benchmark. $\hat{(\cdot)}$ indicates that the parameter is trainable, and "cat" refers to the use of a parameter space decomposition strategy. In SuLoRA, A is frozen, while B uses the parameter space decomposition strategy. The best results are in **bold** and the second-best results are underlined.

that our approach remains effective and feasible even when faced with different backbones and diverse tasks. At the same time, it also validates the effectiveness of our parameter separation strategy. Separating the parameter space for different tasks indeed enhances model performance without increasing the number of training parameters.

5.2 Results on Domain generalization tasks

To further validate the generalization capability of SuLoRA, we conducted additional tests on Domainbed (Gulrajani and Lopez-Paz, 2020), and the experimental results are shown in Table 2. As expected, SuLoRA achieved the best overall performance among all methods. Compared to AsyLoRA, SuLoRA showed an average improvement of **2.50%** and **1.81%** in ID and OOD, respectively. The results on Domainbed further demonstrate that our method remains effective even when tackling domain generalization tasks in computer vision.

5.3 Results on Multi-modal tasks

To evaluate SuLoRA's performance on multi-modal tasks, we tested it on various multi-modal tasks, as shown in Table 4. It can be observed that, when experiments are conducted with the same rank, SuLoRA consistently outperforms LoRA and AsyLoRA, achieving improvements of **5.18%** and **1.75%** in average accuracy compared to the two methods, respectively. The superior performance in multi-modal tasks further demonstrates the robustness and effectiveness of SuLoRA when dealing with different tasks.

6 Analysis

6.1 Analysis of Different Design Strategies for SuLoRA

In this section, we will delve into the impact of different design strategies. We conducted experiments on various benchmarks, and the results on GLUE and INSTRUCTEVAL are shown in Table 5 and Table 6, respectively.

The design applied in SuLoRA (i.e., freezing A and partitioning the subspace of B) consistently achieved superior or comparable performance compared to other design strategies across different benchmarks.

- Compared to $\hat{\mathbf{B}}_{\text{cat}}\hat{\mathbf{A}}$ (i.e., the SuLoRA where both A and B are trained), SuLoRA achieved better performance on a significant number of datasets, especially on INSTRUCTEVAL, while reducing the trainable parameters by half. This suggests that freezing the A matrix can indeed reduce the number of trainable parameters without excessive loss, and even improve model performance. Additionally, freezing A can provide better generalization, further supporting the notion that the A matrix in LoRA is closely related to generalization ability (Zhu et al., 2024).
- Compared to $\hat{\mathbf{B}}\hat{\mathbf{A}}_{\text{cat}}$ and $B\hat{\mathbf{A}}_{\text{cat}}$, SuLoRA demonstrates a clear advantage, indicating that applying the subspace strategy to B is more effective than applying it to A. We believe this is because A is responsible for extracting shared features, and the parameter space activations for shared features across different tasks are highly similar.

Method	N	CoLA	SST-2	MRPC	QNLI	STS-B	RTE	MNLI	QQP	Avg.
SuLoRA	4	66.81	95.53	87.50	93.25	89.66	71.11	89.04	90.57	85.43
SuLoRA	6	71.05	96.26	90.88	93.91	91.26	86.54	89.77	90.91	88.75
SuLoRA	8	71.22	**96.55**	**91.02**	94.12	**91.84**	87.01	90.06	91.51	**89.17**
SuLoRA	10	71.15	95.53	91.06	94.15	91.78	86.96	90.08	91.50	89.03
SuLoRA	12	**71.26**	95.64	**91.02**	**94.19**	**91.84**	**87.10**	**90.12**	**91.53**	89.09

Table 7: **The performance of SuLoRA under different parameter configurations.** The experiments are conducted on the GLUE benchmark. In the experiment, the rank of SuLoRA is set to 4. When $N = r = 4$, SuLoRA degenerates into AsyLoRA. **Boldface** highlights the best results according to the corresponding metrics, while the second-best results are underlined.

Therefore, there is no need to apply the subspace strategy to A. Additionally, we found that training B is more effective than not training it, further supporting the idea that the B matrix in LoRA is indeed associated with task-specific capabilities. Thus, implementing parameter space separation for LoRA's B matrix to enhance model performance is logically sound.

6.2 Analysis of the Number of Parameter Subspaces

We analyze the number of parameter subspaces, N. As shown in Table 7, we find that SuLoRA achieves the best performance when $N = 2r$. When $N > 2r$, the benefits of increasing the number of subspaces begin to diminish, and in some datasets, performance even deteriorates. We believe this is because, although the parameter space separation strategy allocates different subspaces to different instances, when the number of candidate subspaces increases significantly, there is a risk of selecting poorly trained subspaces, which leads to performance degradation on certain datasets.

6.3 Analysis of the Rank of SuLoRA

To analyze the impact of r, we conducted experiments on the performance of SuLoRA with different r values on GLUE, and the results are shown in Fig. 3. As r increases, the overall performance of the model tends to improve. This observation suggests that when there is sufficient training data, a higher rank and more trainable parameters are generally favored in terms of performance. However, higher rank and more trainable parameters usually imply higher training costs. For a comparison with LoRA and more detailed results, please refer to the Appendix D.1.

7 Conclusion

In this paper, we propose a novel parameter-efficient fine-tuning method, SuLoRA. This

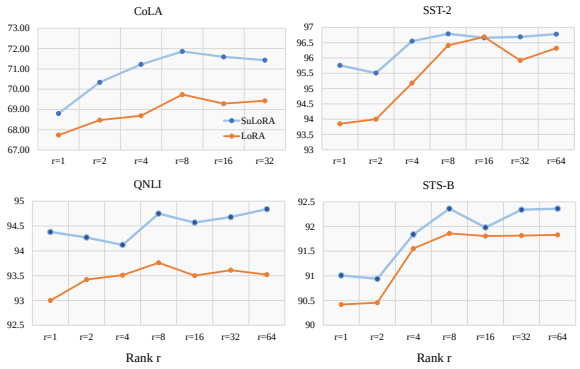


Figure 3: Performance of SuLoRA with different ranks and fixed $N = 2r$ on different datasets.

method adopts an innovative strategy that dynamically selects the optimal parameter subspace based on specific instances and tasks, enabling the separation of parameter spaces across different instances and tasks. This effectively reduces parameter interference between tasks and improves fine-tuning efficiency. Through extensive experiments on multiple benchmarks, we fully validate the effectiveness of SuLoRA's parameter space segmentation strategy. Compared to the traditional LoRA with shared parameter spaces, SuLoRA not only demonstrates stronger adaptability but also shows significant advantages in alleviating parameter interference.

Limitations

Our study focuses on allocating different parameter subspaces for each task, while designing dynamic subspace numbers for each task and fine-tuning in the field of computer vision are left as future research directions. Additionally, due to training cost concerns, our experiments did not further use Llama2-13b to validate the effectiveness of our method on larger LLMs. Therefore, future research could validate the effectiveness of our method on more LLMs and instruction-tuning datasets. Moreover, compared to traditional LoRA, our method

introduces additional computational overhead due to the dynamic parameter subspace selection.

Acknowledgment

This work was funded by the National Natural Science Foundation of China under Grant No.U21B2048 and No.62302382, Shenzhen Key Technical Projects under Grant CJGJZD2022051714160501, China Postdoctoral Science Foundation No.2024M752584.

References

Shivaji Alaparthi and Manit Mishra. 2020. Bidirectional encoder representations from transformers (bert): A sentiment analysis odyssey. *arXiv preprint arXiv:2007.01127*.

Roy Bar-Haim, Ido Dagan, and Idan Szpektor. 2014. Benchmarking applied semantic inference: The PASCAL recognising textual entailment challenges. In *Language, Culture, Computation. Computing - Theory and Technology*, volume 8001 of *Lecture Notes in Computer Science*, pages 409–424. Springer.

Emmanuel Bengio, Joelle Pineau, and Doina Precup. 2020. Interference and generalization in temporal difference learning. In *International Conference on Machine Learning*, pages 767–777. PMLR.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009a. The fifth pascal recognizing textual entailment challenge. *TAC*, 7:8.

Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. 2009b. The fifth PASCAL recognizing textual entailment challenge. In *TAC*. NIST.

Tom B. Brown, Benjamin Mann, Nick Ryder, et al. 2020. Language models are few-shot learners. In *NeurIPS*.

Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity - multilingual and cross-lingual focused evaluation. *CoRR*, abs/1708.00055.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, et al. 2021. Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

Yew Ken Chia, Pengfei Hong, Lidong Bing, and Soujanya Poria. 2023. Instructeval: Towards holistic evaluation of instruction-tuned large language models. *arXiv preprint arXiv:2306.04757*.

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

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *IJCNLP-IWP*. Asian Federation of Natural Language Processing.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *NAACL*.

Chen Fang, Ye Xu, and Daniel N. Rockmore. 2013. Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. In *2013 IEEE International Conference on Computer Vision*, pages 1657–1664.

Chin-Lun Fu, Zih-Ching Chen, Yun-Ru Lee, and Hung-yi Lee. 2022. Adapterbias: Parameter-efficient token-dependent representation shift for adapters in nlp tasks. *arXiv preprint arXiv:2205.00305*.

Xinyuan Gao, Songlin Dong, Yuhang He, Qiang Wang, and Yihong Gong. 2024. Beyond prompt learning: Continual adapter for efficient rehearsal-free continual learning. In *European Conference on Computer Vision*, pages 89–106. Springer.

Sina Ghiassian, Banafsheh Rafiee, Yat Long Lo, and Adam White. 2020. Improving performance in reinforcement learning by breaking generalization in neural networks. *arXiv preprint arXiv:2003.07417*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *ACL-PASCAL*, pages 1–9, Prague. Association for Computational Linguistics.

Ishaan Gulrajani and David Lopez-Paz. 2020. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*.

Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jia-Wei Low, Lidong Bing, and Luo Si. 2021. On the effectiveness of adapter-based tuning for pretrained language model adaptation. *arXiv preprint arXiv:2106.03164*.

Dan Hendrycks, Collin Burns, et al. 2020. Measuring massive multitask language understanding. In *ICLR*.

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*.

Alex Krizhevsky et al. 2009. Learning multiple layers of features from tiny images.

Yann LeCun. 1998. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

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. *arXiv preprint arXiv:2104.08691*.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. 2017. Deeper, broader and artier domain generalization. In *International Conference on Computer Vision*.

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

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Fangyu Liu, Guy Emerson, and Nigel Collier. 2022. Visual spatial reasoning. *CoRR*, abs/2205.00363.

Vincent Liu, Han Wang, Ruo Yu Tao, Khurram Javed, Adam White, and Martha White. 2023. Measuring and mitigating interference in reinforcement learning. In *Conference on Lifelong Learning Agents*, pages 781–795. PMLR.

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

Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 364.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392. The Association for Computational Linguistics.

Pengjie Ren, Chengshun Shi, Shiguang Wu, Mengqi Zhang, Zhaochun Ren, Maarten de Rijke, Zhumin Chen, and Jiahuan Pei. 2024. Mini-ensemble low-rank adapters for parameter-efficient fine-tuning. *arXiv preprint arXiv:2402.17263*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Ying Shen, Zhiyang Xu, Qifan Wang, Yu Cheng, Wenpeng Yin, and Lifu Huang. 2024. Multimodal instruction tuning with conditional mixture of lora. *arXiv preprint arXiv:2402.15896*.

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 *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 8317–8326. Computer Vision Foundation / IEEE.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642. ACL.

Aarohi Srivastava, Abhinav Rastogi, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Aarohi Srivastava, Abhinav Rastogi, et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

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*.

Hugo Touvron, Louis Martin, Kevin Stone, et al. 2023a. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Hugo Touvron, Louis Martin, Kevin Stone, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. 2017. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5018–5027.

Alex Wang. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Xujia Wang, Haiyan Zhao, Shuo Wang, Hanqing Wang, and Zhiyuan Liu. 2024. Malora: Mixture of asymmetric low-rank adaptation for enhanced multi-task learning. *arXiv preprint arXiv:2410.22782*.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*, pages 1112–1122. Association for Computational Linguistics.

Wenhan Xia, Chengwei Qin, and Elad Hazan. 2024. Chain of lora: Efficient fine-tuning of language models via residual learning. *CoRR*, abs/2401.04151.

Zhiyang Xu, Trevor Ashby, Chao Feng, Rulin Shao, Ying Shen, Di Jin, Qifan Wang, and Lifu Huang. 2023. Vision-flan: Scaling visual instruction tuning.

Hantao Yao, Rui Zhang, and Changsheng Xu. 2024. Tcp: Textual-based class-aware prompt tuning for visual-language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23438–23448.

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

Jiacheng Zhu, Kristjan Greenewald, Kimia Nadjahi, Haitz Sáez De Ocáriz Borde, Rickard Brüel Gabrielsson, Leshem Choshen, Marzyeh Ghassemi, Mikhail Yurochkin, and Justin Solomon. 2024. Asymmetry in low-rank adapters of foundation models. In *Proceedings of the 41st International Conference on Machine Learning*, pages 62369–62385.

A Additional Experimental Details

A.1 Additional Benchmark Details

Natural Language Understanding (NLU) Task. The GLUE benchmark is used for NLU tasks, including classification, semantic similarity, paraphrase, and natural language inference tasks. It includes the CoLA (Xia et al., 2024), SST-2 (Socher et al., 2013), MRPC (Dolan and Brockett, 2005), QNLI (Rajpurkar et al., 2016), STS-B (Cer et al., 2017), RTE (Bentivogli et al., 2009b; Bar-Haim et al., 2014; Giampiccolo et al., 2007; Bentivogli et al., 2009a), MNLI (Williams et al., 2018), and QQP datasets. The detailed information for each benchmark and dataset is provided in Table 8.

Dataset	#Train	#Dev	#Test	#Metrics
GLUE BM				
CoLA	8.5k	1k	1k	Matthews corr
SST-2	67k	872	1.8k	Accuracy
MRPC	3.7k	408	1.7k	AccuracyF1
QNLI	108k	5.7k	5.7k	Accuracy
STS-B	7k	1.5k	1.4k	Pearson
RTE	2.5k	276	3k	Accuracy
MNLI	393k	20k	20k	Accuracy
QQP	364k	40k	391k	AccuracyF1
INSTRUCT EVAL BM				
MMLU	99.8k	1.8k	14k	Accuracy
DROP	77.4k	0	9.5k	Accuracy
HEval	0	0	164	pass@1
BBH	0	0	6.5k	Accuracy

Table 8: Datasets Information. "BM" is short for "Benchmark". We train the models on the cleaned Alpaca dataset and evaluate them on the test sets of INSTRUCTEVAL.

- **CoLA** (Xia et al., 2024) dataset consists of English acceptability judgments collected from linguistic theory sources, aiming to evaluate models' ability to distinguish grammatically correct sentences from incorrect ones.
- **SST-2** (Socher et al., 2013) dataset consists of movie review sentences with human-annotated sentiment labels, designed for binary sentiment classification tasks.
- **MRPC** (Dolan and Brockett, 2005) dataset consists of sentence pairs extracted from online news sources, designed to evaluate whether two sentences are semantically equivalent.
- **QNLI** (Rajpurkar et al., 2016) dataset is a question-answering dataset consisting of question-paragraph pairs, designed to determine whether a context sentence contains the answer to a given question.
- **STS-B** (Cer et al., 2017) dataset consists of sentence pairs drawn from news headlines, video and image captions, and natural language inference data, with human-annotated similarity scores ranging from 1 to 5, designed to predict these similarity scores.
- **RTE** (Bentivogli et al., 2009b; Bar-Haim et al., 2014; Giampiccolo et al., 2007) datasets consist of challenges from a series of annual textual entailment tasks, designed to predict whether a premise entails a given hypothesis.
- **MNLI** (Williams et al., 2018) dataset is a crowdsourced collection of sentence pairs with textual entailment annotations, designed to assess whether a premise entails, contradicts, or is neutral with respect to a given hypothesis.
- **QQP** dataset consists of question pairs from the Quora[1] community question-answering website, aiming to determine whether the two questions are semantically equivalent.

Instruction Following Tasks. We fine-tuned Llama2-7b (Touvron et al., 2023b) with our SuLoRA on the cleaned Alpaca data and evaluated its performance on the instruction-following task using the INSTRUCTEVAL (Chia et al., 2023) benchmark, which includes the MMLU (Hendrycks et al., 2020), DROP (Dua et al., 2019), HumanEval (HEval) (Chen et al., 2021), and BBH (Srivastava et al., 2023) datasets.

- **MMLU** (Hendrycks et al., 2020) dataset is designed to assess world knowledge and problem-solving abilities across a broad range of subjects, evaluating models in both zero-shot and few-shot settings, with questions ranging from elementary to advanced professional levels.
- **DROP** (Dua et al., 2019) dataset is a math-based reading comprehension task that requires systems to perform discrete reasoning on passages extracted from Wikipedia articles, involving operations such as addition, counting, and sorting.
- **HumanEval** (HEval) (Chen et al., 2021) is a benchmark used to evaluate the problem-solving abilities of large language models in programming. It consists of 164 original programming problems that assess language comprehension, algorithms, and basic mathematics, with some problems comparable to simple software interview questions.
- **BBH** (Srivastava et al., 2023) is a subset of 23

[1] https://huggingface.co/datasets/glue/viewer/qqp

challenging tasks from the BIG-Bench (Srivastava et al., 2022) benchmark, focusing on tasks that are believed to be beyond the capabilities of current language models. It requires models to perform challenging instructions such as navigation, logical deduction, and fallacy detection.

Domain generalization tasks. In addition, to further validate the effectiveness of our method, we also conducted visual domain generalization tests on the Domainbed benchmark, which includes the VLCS (Fang et al., 2013), PACS (Li et al., 2017), and OfficeHome (Venkateswara et al., 2017) datasets. The training and testing data are split following the original 80% training and 20% testing ratio.

- **VLCS** (Fang et al., 2013) is an important resource for domain generalization tasks. The goal of domain generalization is to train a model that performs well on unseen domains, even when their data distributions differ from the training data.
- **PACS** (Li et al., 2017) is a domain adaptation image dataset that contains four domains: photographs (1,670 images), art paintings (2,048 images), cartoons (2,344 images), and sketches (3,929 images), with each domain consisting of 7 categories. The dataset is divided into three parts: the training set with 8,977 images, the test set with 1,014 images, and the validation set with 9,991 images.
- **Office-Home** (Venkateswara et al., 2017) is a benchmark dataset for domain adaptation, consisting of 4 domains, each containing 65 categories with an average of 70 images per category, totaling 15,500 images. The four domains and their details are as follows: Art, which includes artistic images in the form of sketches, paintings, and decorations; Clipart, a collection of clipart images; Product, featuring images of objects without backgrounds; and Real-World-images, containing images of objects captured by regular cameras.

Multimodal tasks. To verify the generalization of our method, we follow (Shen et al., 2024) and perform instruction tuning on a Vision-Flan (Xu et al., 2023) subset, a multimodal dataset with 187 tasks. The subset includes up to 1,000 instances per task, totaling 182,167 instances. Subsequently, we test on four multimodal datasets: Text-VQA (Singh et al., 2019), VSR (Liu et al., 2022), CIFAR-100 (Krizhevsky et al., 2009), and MNIST (LeCun, 1998). Below are the detailed explanations of these four datasets.

- **Text-VQA** (Singh et al., 2019) is a dataset for Visual Question Answering (VQA), where the goal is to answer questions related to an image that require understanding both visual content and textual information. The dataset contains images paired with questions and answers, and these questions often involve text within the images, requiring models to integrate both visual and textual understanding.
- **VSR** (Liu et al., 2022) is a dataset designed for recognizing sentiment in images based on both visual and textual cues. It contains images annotated with sentiment labels (such as positive, negative, or neutral) along with related textual descriptions. The task is to classify the sentiment of the image based on its visual content and any accompanying text.
- **CIFAR-100** (Krizhevsky et al., 2009) is a well-known image classification dataset containing 60,000 32x32 color images across 100 classes. The dataset is divided into 50,000 training images and 10,000 testing images. It is commonly used for evaluating image classification models, where each image is labeled with one of the 100 classes, ranging from animals to objects.
- **MNIST** (LeCun, 1998) dataset is a collection of handwritten digits from 0 to 9. It consists of 60,000 training images and 10,000 testing images, each image being a 28x28 grayscale image of a digit. MNIST is widely used for benchmarking image recognition models, particularly in the area of digit recognition.

A.2 Additional Introduction to Comparison Methods

Full finetuning (FT) - the method is initialized with pretrained weights, and all parameters are involved in training.

LoRA (Hu et al., 2021) - as introduced in the section 2.

AdaLoRA (Zhang et al., 2023) - the method focuses on determining the optimal rank for each layer in the model. It adjusts the rank selection based on the magnitude of each singular value.

MELoRA (Ren et al., 2024) - the method focuses on stacking multiple mini LoRAs in parallel. It concatenates multiple mini LoRAs along the diagonal to construct an equivalent block-diagonal LoRA matrix.

AsyLoRA (Zhu et al., 2024) - as introduced in section 2.

B Further Explanation of SuLoRA

B.1 A More Detailed Motivation Explanation

From the perspective of gradients, we agree that gradient differences inherently capture task-specific features. However, within a shared parameter space, differential gradients may introduce harmful interference due to conflicting update directions, as illustrated in Figure 1(b) of the main text.

- **Parameter Preference Divergence:** Our Equation (3) and Figure 1(b) demonstrate that the LoRA B matrix subspaces activated by different tasks exhibit high divergence. When parameters are shared, these conflicting gradients are forced to optimize the same parameters, resulting in overwriting or cancellation.

- **Evidence of Performance Degradation:** Tables 1 and 7 show that SuLoRA with separated parameters significantly outperforms LoRA with shared parameters, especially in cross-task scenarios (e.g., a 3.07% improvement on the RTE task), confirming the presence of interference.

Therefore, although gradient differences naturally exist, they must be managed through parameter isolation (e.g., subspace allocation in SuLoRA) to prevent them from turning into harmful conflicts. In a shared parameter space, differentiated gradients can overwrite or cancel each other out, analogous to negative transfer in multitask learning, ultimately degrading the model's adaptability to specific tasks.

From the perspective of tasks and instances, our goal is to address subtasks within datasets (e.g., classes). We associate different classes with different tasks. For example, in Figure 1(b), different classes are treated as distinct tasks. The instance-level routing mechanism in SuLoRA is essentially a finer-grained solution for task-level parameter isolation; the two approaches are not contradictory but complementary. By dynamically allocating instance-level subspaces, more flexible parameter isolation across tasks can be achieved.

B.2 More Explanation of Parameter Grouping

The following are explanations regarding parameter group partitioning and activation/sleep thresholds:

1. **Parameter Group Partitioning.**
   The core concept of parameter grouping in Section 3.2 involves structurally decomposing the LoRA matrix B into multiple parameter groups. For example, a $d \times r$ matrix B can be divided into r column vectors of size $d \times 1$, where each column vector represents an independent d-dimensional parameter group. Physically, this corresponds to partitioning matrix B column-wise, where each parameter group (or subspace) corresponds to a latent task-specific feature pattern.

2. **Activation/Sleep Thresholds.**
   Thresholds of 0.7 (activation) and 0.3 (sleep) are determined based on statistical distributions. During fine-tuning on STS-B, we computed scores for the partitioned parameter groups and set these thresholds according to the mean and standard deviation of the final statistics.

B.3 Training Mechanism Clarification

We adopt a load balancing strategy similar to MoE. SuLoRA draws on the load balancing concept from MoE and employs an entropy maximization constraint to ensure comprehensive subspace training. Specifically, we add a load balancing loss to the training objective, which maximizes the entropy of subspace selection to encourage the router to explore underutilized subspaces.

Formally, let the probability distribution over subspace selections be $p = (p_1, p_2, \ldots, p_K)$, where K is the number of subspaces. The load balancing loss is defined as the negative entropy of this distribution:

$$\mathcal{L}_{\text{load}} = -\sum_{k=1}^{K} p_k \log p_k. \tag{6}$$

By minimizing this loss, we maximize the entropy, encouraging balanced utilization of all subspaces during training.

B.4 Analysis of Parameter Space Arrangement

First, we analyzed the architecture of SuLoRA itself. After selecting and merging subspaces, SuLoRA optimizes the combined parameter space as a whole, which to some extent mitigates the issue of individual subspaces being overly specialized in a single type of feature. Second, to validate

Method	Random	SST-2	RTE	QQP
LoRA	-	95.57	85.32	91.95
SuLoRA r=8	Y	96.78	88.32	92.23
SuLoRA (Ours) r=8	N	96.79	88.39	92.20

Table 9: Subspace Arrangement Analysis

SuLoRA's effectiveness in mitigating permutation inconsistency, we conducted a robustness analysis experiment involving subspace order permutation during the testing phase. In this experiment, after selecting the subspace indices for testing, we randomly permuted the chosen subspaces and performed inference based on the shuffled configuration. Below we present the robustness test results, with all experiments conducted on the GLUE benchmark. The detailed experimental results are shown in Table 9.

C Additional Experiment

C.1 More Detailed Experiments on Multimodal Tasks

Method	Text-VQA	VSR	CIFAR-100	MNIST	MME
LoRA $_{r=8}$	39.20	53.27	46.88	82.95	1312.87
MELoRA	<u>40.72</u>	53.65	57.24	85.19	<u>1430.97</u>
AsyLoRA	40.63	<u>52.81</u>	57.29	85.32	1422.05
SuLoRA	**42.31**	**53.92**	<u>60.22</u>	<u>86.57</u>	**1487.60**

Table 10: **More detailed experiments on multimodal tasks.** The best results are in **bold** and the second-best results are <u>underlined</u>.

We present the performance on multimodal tasks, including MME, in Table 10, to further demonstrate the effectiveness of our method in multimodal settings.

D Additional Analysis

D.1 Additional Analysis of the Rank of SuLoRA

In this section, we present the impact of different ranks on SuLoRA across the remaining datasets in GLUE. As shown in Fig. 4, SuLoRA outperforms LoRA across all rank settings.

D.2 Computational Cost Analysis

We conducted a comparative analysis of memory consumption, computation speed, and performance with different numbers of subspaces (N). These experiments were performed on the CoLA dataset, and the results are shown in the Table 11. An asterisk ($*$) indicates the hyperparameters we used.

Figure 4: Performance of SuLoRA with different ranks and fixed $N = 2r$ on different datasets.

Method	N	Memory	it/s	CoLA
MELoRA	-	12591	1.42	70.67
SuLoRA (Ours) r=8	12	13070	1.59	71.49
SuLoRA (Ours) r=8 *	16	13076	1.45	71.86
SuLoRA (Ours) r=8	20	13082	1.19	71.90

Table 11: Cost Comparison of Different Methods

D.3 Further comparison with HydraLoRA/MALoRA

To further illustrate the differences between our method and multi-head LoRA approaches such as MALoRA (Wang et al., 2024) and HydraLoRA (Tian et al., 2024), we provide a detailed comparison between SuLoRA and these methods in Table 12.

Method	Router Mechanism	Subspace Sharing
SuLoRA	Dynamic routing; selects subspaces	Partial sharing of LoRA matrices
HydraLoRA/MALoRA	Dynamic routing; selects multiple LoRAs	Shares and fuses multiple LoRAs

Computational Efficiency	Suitable Scenarios	Standard MoE?
High (selective activation)	Resource-constrained	No (output order may vary)
Moderate (parallel branch computation)	Compute-rich environments	Yes (follows standard MoE aggregation)

Table 12: Comparison of SuLoRA and HydraLoRA/MALoRA methods (split view)