SLoRA: Shared Low-Rank Adaptation for Parameter-Efficient Fine-Tuning of Large Language Models

Qingjun Zhang , Feng Ji*, Ruisheng Ran
College of Computer and Information Science, Chongqing Normal University, Chongqing, China
Email: {2023110516070, jifeng, rshran}@cqnu.edu.cn

***Abstract*—Low-Rank Adaptation serves as the predominant framework for parameter-efficient fine-tuning of large language models. Despite its effectiveness, standard implementations introduce considerable storage and parameter requirements that may be prohibitive in resource-constrained environments. This work introduces Shared Low-Rank Adaptation, an approach designed to facilitate global parameter sharing across homogeneous modules situated in distinct layers. The methodology involves the construction of a shared weight pool dedicated to modules of identical types. Through the application of selection and splice operators, the framework synthesizes layer-specific matrix pairs characterized by diverse effective ranks derived from the common pool. This architectural design substantially diminishes the total parameter budget while maintaining robust adaptation performance. Comprehensive empirical evaluations indicate that the proposed method achieves a more favorable balance between parameter economy and fine-tuning efficacy than previous variants.**

***Index Terms*—LLMs, Fine-Tuning, LoRA, Weight sharing.**

I. Introduction

Large Language Models (LLMs) have fundamentally transformed the landscape of Natural Language Processing (NLP) [1], [2], demonstrating exceptional generalization capabilities across a wide range of tasks. However, due to the discrepancy between pre-training data distributions and specific downstream domains, practitioners often find it necessary to fine-tune these models to align with specific requirements. To mitigate the prohibitive resource costs associated with Full Fine-Tuning, researchers have developed a series of Parameter-Efficient Fine-Tuning (PEFT) techniques. Among these, Low-Rank Adaptation (LoRA) [3] has been widely adopted for its ability to achieve a compelling trade-off between performance and efficiency. Despite its success in reducing video memory requirements compared to full fine-tuning, the memory overhead associated with storing trainable parameters and optimizer states remains a non-negligible constraint, particularly when deploying large-scale models. Consequently, recent efforts have focused on further optimizing LoRA by reducing the number of trainable parameters without compromising model performance.

*Corresponding author.

This work was supported by Scientific and Technological Research Program of Chongqing Municipal Education Commission (Grant No. KJZD-M202300502).

Recent studies have explored architectures that implement weight or vector sharing across different layers to address this challenge. Methods such as VeRA [4] and VB-LoRA [5] employ mechanisms like frozen random matrices or vector-only training to compress the parameter space. While these approaches significantly reduce the number of trainable parameters, they often impose rigid constraints on the weight update matrices or rely on uniform sharing strategies that may neglect the layer-wise heterogeneity of feature extraction. Such limitations can restrict the model's capacity to adaptively capture complex patterns in diverse layers.

Building on existing works, this paper introduces **S**hared **Lo**w-**R**ank **A**daptation (**SLoRA**). Different from previous methods, SLoRA uses a concise weight-sharing strategy with a global shared pool, layer-wise selection and concatenation. It splices high-rank and low-rank components to form varied effective ranks per layer, better balancing small trainable parameters and fine-tuning performance.

The main contributions of this work are summarized as follows: (1) A parameter-efficient framework, SLoRA, is proposed to better balance the trade-off between fine-tuning performance and memory requirements. (2) A simplified weight-sharing strategy is introduced to reduce the complexity of sharing decisions and facilitate easier implementation. (3) The structure of high-rank concatenation positions and the distribution of splicing factors are analyzed, providing empirical references for further optimization of parameter-efficient methods. The source code of SLoRA is available at: https://github.com/DDYYWWJJ/SLoRA.

II. Related Work

A. Low-Rank Adaptation and Memory-Efficient Variants

Low-Rank Adaptation has been widely adopted for parameter-efficient fine-tuning of large language models. This method decomposes the weight update matrix into the product of two low-rank matrices and maintains the pre-trained weights W_0 frozen, which introduces minimal trainable parameters and achieves model adaptation without incurring additional inference latency. Formally, given a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, the fine-tuning process updates weights via $W = W_0 + \Delta W = W_0 + BA$, where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ denote low-rank matrices with rank $r \ll \min(d, k)$. The

optimization is restricted to matrices A and B to significantly reduce the computational budget.

LoRA-FA [6] reduces activation memory consumption by freezing the down-projection matrix while exclusively optimizing the up-projection matrix. This strategy lowers video memory requirements and preserves model performance comparable to the original adaptation method.

B. Shared and Global Low-Rank Adaptation Variants

Certain LoRA variants investigate parameter sharing mechanisms across layers or modules to reduce trainable parameters and enhance computational efficiency. VB-LoRA [5] introduces a Vector Bank framework that decomposes low-rank matrices into sub-vectors retrieved from a shared pool. By employing a differentiable top-k mechanism to facilitate global sharing among layers and modules, this architecture achieves high-ratio parameter compression.

Vector-based Random Matrix Adaptation, or VeRA [4], proposes sharing a unified pair of global low-rank matrices across all network layers while learning only layer-specific scaling vectors b_l and d_l. This strategy replaces the independent matrices in standard LoRA with shared bases and specific scaling factors. The weight adaptation process is formally approximated as $W_l = W_{0,l} + (BA) \odot \text{diag}(b_l) \odot \text{diag}(d_l)$.This framework significantly lowers cross-layer redundancy and storage requirements while preserving model adaptation capacity.

C. Transformation Structure and Decoupling Strategy

DoRA [7] decomposes weight updates into magnitude and directional components, applying low-rank adaptation exclusively to the direction while optimizing magnitude independently to decouple scale learning and enhance stability. Within the LyCORIS framework, variants such as LoHa [8] and LoKr [8] utilize Hadamard and Kronecker products for weight construction. LoHa employs element-wise multiplication to increase expressivity with minimal parameter overhead, whereas LoKr leverages block-matrix operations to preserve rank properties for efficient high-dimensional approximation.

III. Method

The SLoRA framework establishes a shared weight pool to achieve efficient compression of fine-tuning parameters. During the forward propagation phase, the proposed method executes a strategy that configures splicing factors and employs a dual splicing mode for high-rank and low-rank components. This process integrates the specific positioning of high-rank feature splicing to optimize the representation. Figure 1 illustrates the overall implementation workflow.

A. Global Subspace

SLoR is predicated on the hypothesis that there exists a globally shared subspace across homogeneous weight parameters (e.g., W_Q, W_K, W_V, W_O). Formally, we define this subspace as:

$$\mathcal{U} \subset \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}, \quad \text{where } \dim(\mathcal{U}) \ll d_{\text{out}}, d_{\text{in}} \tag{1}$$

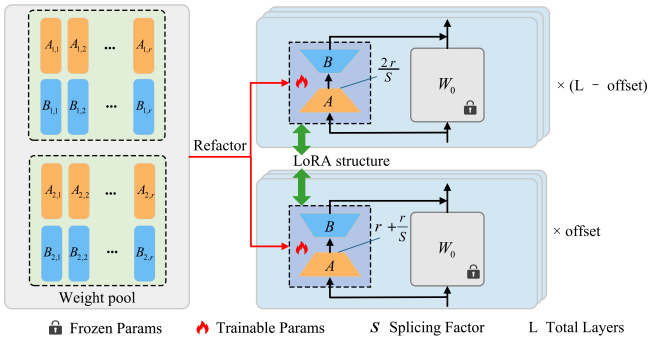

Fig. 1. Schematic illustration of the proposed SLoRA method. The architecture constructs LoRA modules with varying ranks by concatenating shared weights across different layers.

For any arbitrary layer l, the corresponding fine-tuning weight increment, denoted as ΔW_l, is constrained within this subspace ($\Delta W_l \in \mathcal{U}$). Specifically, ΔW_l represents the update to the pre-trained weights and is formulated as the matrix product of A and B, which are constructed by concatenating vectors selected from the shared subspace $\mathcal{U}$.

B. Weight Pool Setup

To facilitate the selection of candidate vectors, we construct a parameter-sharing pool for SLoRA. The capacity of this pool is commensurate with the aggregate dimensions of the A and B matrices derived from the two original LoRA instances. Formally, the sharing pools U and V are defined as:

$$U = \{\mathcal{A}_1^1, \mathcal{A}_1^2, \ldots, \mathcal{A}_1^r, \mathcal{A}_2^1, \mathcal{A}_2^2, \ldots, \mathcal{A}_2^r\} \tag{2}$$

$$V = \{\mathcal{B}_1^1, \mathcal{B}_1^2, \ldots, \mathcal{B}_1^r, \mathcal{B}_2^1, \mathcal{B}_2^2, \ldots, \mathcal{B}_2^r\} \tag{3}$$

where each vector denotes a column slice such that $\mathcal{A}_k^j \in \mathbb{R}^{d_{in}}$ and $\mathcal{B}_k^j \in \mathbb{R}^{d_{out} \times 1}$. Following the standard LoRA initialization protocol, we initialize the weight matrices for these groups as follows: the $\mathcal{B}$ matrix is initialized to zero, whereas the $\mathcal{A}$ matrix follows a random Gaussian initialization, denoted as $\mathcal{A} \sim \mathcal{N}(0, \sigma^2)$.

C. Cross-Layer Weight Sharing

During the fine-tuning stage, SLoRA dynamically reconstructs the weights of specific layers via a parameter selection and splicing mechanism. The forward pass incorporating parameter updates is formulated as follows:

$$h = W_l x + \Delta W_l x = W_l x + \Phi_l(\mathcal{U}) \cdot \Phi_l(\mathcal{V}) x \tag{4}$$

where W_l denotes the frozen pre-trained weights, and $\mathcal{U}, \mathcal{V}$ represent the globally shared parameter pools. The term Φ_l is defined as the selection and splicing operator, which extracts and reassembles basis vectors from the shared subspace based on the constraints of hyperparameters L and S to construct layer-specific low-rank matrices.The hyperparameter L identifies the positions for high-rank splicing. In this configuration, the parameters of the current layer are fused with those of

the adjacent layer $L + \text{offset} - 1$, where the offset is set to 2. The factor S serves as a splicing scaling factor to quantitatively control the dimensions of the vectors involved in the splicing process.The selection and splicing operator follows a consistent logic for both A_l and B_l. To maintain conciseness, the following derivation utilizes matrix A as the primary illustrative case. By treating all layers as a circular queue, higher-rank weight splicing is executed when the layer index matches the designated hyperparameter L. The resulting weight is defined as:

$$A_L = \left[\mathcal{A}_1^{(1:r)}, \mathcal{A}_2^{(1:c)}\right] \in \mathbb{R}^{d_{in} \times (r+c)} \tag{5}$$

For the subsequent layer $L + 1$, the formulation is:

$$A_{L+1} = \left[\mathcal{A}_1^{(r-c:r)}, \mathcal{A}_2^{(1:r)}\right] \in \mathbb{R}^{d_{in} \times (r+c)} \tag{6}$$

In these equations, c represents the quotient of the predefined rank r and the splicing factor S, specifically $c = r/S$. This parameter determines the number of additional basis vectors incorporated into the splicing, thereby extending the effective rank of the current layer from r to $r + c$.For layers not designated by L, a subset of sub-vectors is selected for lower-rank splicing to mitigate distribution shift. The calculation is formulated as:

$$A_{other} = \left[\mathcal{A}_1^{(r-c:r)}, \mathcal{A}_2^{(1:c)}\right] \in \mathbb{R}^{d_{in} \times 2c} \tag{7}$$

Based on the predefined splicing configuration, slicing operations are performed on the source matrices $\mathcal{A}_1$ and $\mathcal{A}_2$ to extract c vector components respectively. These components are ultimately aggregated to yield a restructured LoRA matrix with an equivalent rank of $2c$.

IV. Experiments

A. Experimental Setup

1) Datasets: To thoroughly validate the proposed SLoRA framework, we perform extensive experiments across four capability dimensions as follows: (1) Mathematical Reasoning (Math): We utilized the GSM8K [9] training set for model training and assessed performance on its corresponding test set. (2) Natural Language Understanding (NLU): Training was conducted on an aggregated corpus comprising eight standard benchmarks: BoolQ [10], PIQA [11], SocialIQA [12], ARC-Challenge [13], ARC-Easy [13], OpenBookQA [14], HellaSwag [15], and Winogrande [16]. We reported the accuracy on the independent test sets for each respective dataset. (3) Code Generation (Code): The models were trained on the CodeAlpaca [17] dataset. For evaluation, we employed the HumanEval [18] benchmark and adopted pass@1, pass@5, and pass@10 as the performance metrics. (4) Safety Alignment (Safety): We utilized Saferpaca [19] for training, a dataset derived from the cleaned Alpaca corpus augmented with an additional 2,000 safety-specific instructions. Safety capability was quantified by the refusal rate on harmful queries within the HEx-PHI benchmark [20].

2) Baseline: To comprehensively evaluate the performance of SLoRA, we benchmark it against several mainstream PEFT methods. Specifically, these baselines include: (1) Classical Low-Rank Adaptation methods and their variants, namely LoRA, LoRA-FA, VBLoRA, and DoRA. (2) Methods based on structured parameter reconfiguration, specifically LoKR and LoHA, which originate from the LyCORIS project and are implemented via the Hugging Face PEFT library. To ensure a fair comparison, all baseline models are evaluated under a unified experimental setting.

3) Implementation Details: Experiments employ Llama-3-8B-Instruct [21] and Mistral-7B [22] on four NVIDIA A100-40GB GPUs. We determine the optimal SLoRA configuration by identifying the most effective high-rank splice layer position and splice factor, with a detailed impact analysis provided in Section 4.2. Remaining hyperparameters follow Zhang et al. [23] as listed in Table I.

TABLE I
HYPERPARAMETER CONFIGURATION

params	NLU	Saferpaca	CodeAlpaca	GSM8K
Lora_a	32	64	32	64
Lora_alpha	64	128	64	128
Learning rate	5e-5	1e-5	5e-5	
Dropout	0.05			
Optimizer	AdamW			
Batch size	64			
Warmup steps	0			
Epochs	2			
where	q, v, k, o, gate, up, down			

B. Results

1) Task Performace: Tables II- VI report multi-domain performance and trainable parameter ratios across various rank configurations for all methods. **Bold** indicates the best-performing method, and underline indicates the second-best.

The experimental results demonstrate that SLoRA exhibits robust adaptability across the majority of Natural Language Understanding datasets, consistently achieving either optimal or near-optimal performance. Notably, under experimental settings involving two distinct models and rank configurations, SLoRA attains scores comparable to, and in some instances surpassing, those of LoRA and its variants. It is significant that this performance is achieved despite SLoRA possessing a parameter count only marginally higher than VeRA, which serves as the baseline for minimal parameter usage among the compared methods. These findings substantiate the efficacy of SLoRA in NLU tasks, as it successfully balances parameter efficiency with competitive performance.

In the evaluation of the remaining three tasks, DoRA outperforms other methods across five metrics spanning three domains. However, given that DoRA utilizes a higher parameter budget compared to SLoRA, the observed performance gains do not appear proportional to the increased computational cost. Conversely, SLoRA frequently secures the second-best results or matches the performance of LoRA in the domains

TABLE II
PERFORMANCE OF DIFFERENT METHODS ON NLU TASKS USING LLAMA AND MISTRAL BACKBONES WITH RANKS $r = 32$ AND 64.

Model-Rank	Method	Params(%)	boolQ	PIQA	SIQA	ARC-c	ARC-e	OBQA	Hellas	WinoG	Avg
Llama-32	lora	84M(1.04)	**73.86**	88.28	80.72	**83.15**	92.44	**86.83**	94.75	86.92	85.87
	lorafa	44M(0.54)	73.07	**89.95**	**81.40**	81.85	**92.74**	**86.83**	95.06	86.35	**85.91**
	vera	1M(0.01)	64.79	82.47	63.90	68.48	86.82	61.60	81.11	58.55	70.97
	vblora	20M(0.24)	64.39	83.98	72.86	75.60	88.21	75.66	89.36	73.84	77.99
	lokr	2M(0.02)	69.42	86.77	74.37	77.17	89.86	77.90	91.98	76.56	80.50
	loha	168M(2.08)	70.92	88.11	77.13	79.42	90.79	81.47	93.61	82.15	82.95
	dora	85M(1.06)	73.56	87.27	80.31	82.72	91.59	**86.83**	**95.39**	**87.58**	85.66
	slora	5M(0.06)	72.64	89.67	79.89	81.16	91.72	85.93	94.46	85.93	85.18
Mistral-32	lora	84M(1.04)	71.99	86.88	78.85	79.51	91.17	**86.38**	93.86	85.44	84.26
	lorafa	44M(0.54)	71.93	86.88	**80.57**	77.60	88.81	85.26	93.16	87.33	83.94
	vera	1M(0.01)	64.36	84.54	72.03	73.26	87.45	72.54	88.90	69.98	76.63
	vblora	20M(0.24)	67.52	85.32	75.78	73.78	87.79	77.90	90.72	75.08	79.24
	lokr	2M(0.02)	72.91	88.39	78.54	80.46	**91.25**	84.15	94.85	81.66	84.03
	loha	168M(2.08)	73.43	88.95	79.94	**81.85**	91.21	83.92	**95.19**	83.38	84.73
	dora	85M(1.06)	73.03	87.66	78.07	79.25	91.17	85.71	94.41	84.45	84.22
	slora	5M(0.06)	**73.56**	**89.45**	80.00	81.07	90.75	86.16	**95.19**	**86.26**	**85.31**
Llama-64	lora	84M(1.04)	**72.70**	87.27	79.68	80.98	90.37	87.05	93.96	86.01	84.75
	lorafa	44M(0.54)	72.03	88.39	80.73	80.12	**92.31**	**87.94**	94.78	**87.42**	**85.46**
	vera	1M(0.01)	58.55	82.58	63.95	68.66	87.16	63.17	81.32	58.71	70.51
	vblora	20M(0.24)	64.55	68.69	74.11	75.78	88.93	76.56	89.92	74.75	76.66
	lokr	2M(0.02)	67.64	84.98	72.81	76.04	89.10	75.00	90.58	72.45	78.58
	loha	168M(2.08)	71.17	88.00	78.22	79.68	91.30	82.14	94.08	82.40	83.37
	dora	85M(1.06)	70.80	86.94	79.27	**82.03**	91.08	87.27	93.93	87.41	84.84
	slora	5M(0.06)	71.72	**89.06**	**81.04**	81.16	91.93	86.16	**95.05**	86.26	85.30
Mistral-64	lora	84M(1.04)	**75.20**	89.00	81.20	**82.30**	**92.40**	**89.10**	95.30	**88.20**	**86.60**
	lorafa	44M(0.54)	68.23	83.92	79.37	74.91	85.76	81.25	91.34	83.55	81.04
	vera	1M(0.01)	65.56	84.31	71.88	74.39	88.26	72.09	88.74	69.98	76.90
	vblora	20M(0.24)	67.00	82.97	74.79	70.39	81.12	74.55	74.35	71.05	74.53
	lokr	2M(0.02)	71.69	87.22	77.65	78.90	90.75	82.36	93.73	78.28	82.57
	loha	168M(2.08)	73.40	88.50	80.20	80.46	91.34	85.26	95.39	84.04	84.82
	dora	85M(1.06)	70.06	83.98	75.93	74.21	84.03	80.80	89.99	83.47	80.31
	slora	5M(0.06)	71.13	**89.78**	**81.56**	82.20	91.34	88.16	**95.66**	85.03	85.61

TABLE III
PERFORMANCE COMPARISON ON LLAMA WITH RANK $r = 32$

Method	GSM8K	HumanEval			Hex-PHI
		Pass@1	Pass@5	Pass@10	
lora	55.31	**39.51**	52.20	57.27	**100.0**
lorafa	54.68	34.60	43.57	46.79	**100.0**
vera	51.32	30.70	44.27	49.08	84.37
vblora	51.64	30.67	46.06	52.09	93.75
lokr	49.06	32.65	45.86	51.03	95.31
loha	52.65	38.29	50.74	56.29	98.43
dora	**58.04**	32.16	43.64	48.96	96.87
slora	54.06	39.05	**52.90**	**58.71**	98.43

TABLE IV
PERFORMANCE COMPARISON ON MISTRAL WITH RANK $r = 32$

Method	GSM8K	HumanEval			Hex-PHI
		Pass@1	Pass@5	Pass@10	
lora	50.62	31.92	41.12	44.16	**98.44**
lorafa	47.65	29.51	36.95	41.72	98.43
vera	46.64	29.42	38.26	41.76	89.06
vblora	47.65	27.86	37.39	41.99	85.93
lokr	46.40	30.30	39.86	43.77	96.87
loha	46.71	30.51	39.68	42.90	96.87
dora	**54.84**	**32.62**	41.79	45.19	98.43
slora	50.62	32.34	**41.97**	**46.04**	98.43

TABLE V
PERFORMANCE COMPARISON ON LLAMA WITH RANK $r = 64$

Method	GSM8K	HumanEval			Hex-PHI
		Pass@1	Pass@5	Pass@10	
lora	55.23	35.09	**49.85**	**55.03**	97.81
lorafa	54.84	33.35	41.77	45.13	92.81
vera	52.42	29.02	42.77	48.94	81.25
vblora	52.65	33.01	46.06	51.50	86.25
lokr	47.42	30.34	43.20	47.79	84.68
loha	53.35	31.67	44.64	49.72	89.37
dora	**57.81**	33.07	44.12	49.14	**98.43**
slora	54.84	**35.60**	49.14	54.76	97.18

TABLE VI
PERFORMANCE COMPARISON ON MISTRAL WITH RANK $r = 64$

Method	GSM8K	HumanEval			Hex-PHI
		Pass@1	Pass@5	Pass@10	
lora	49.84	30.82	40.60	44.20	93.75
lorafa	40.93	29.18	36.70	39.66	95.93
vera	47.34	29.32	38.93	43.03	81.25
vblora	44.45	24.02	33.26	37.35	84.06
lokr	42.89	29.02	39.34	43.56	95.00
loha	48.51	29.14	38.74	42.46	97.50
dora	**53.35**	**33.10**	41.84	45.34	92.50
slora	52.18	31.40	**42.28**	**46.65**	**98.12**

of mathematical reasoning and code generation. Regarding safety alignment, SLoRA does not consistently achieve leading metrics. This result suggests that SLoRA's current mechanism leaves room for further optimization, particularly in balancing safety risk mitigation with task performance. This may stem from the possibility that weight sharing partially disrupts specific neurons associated with safety guardrails.

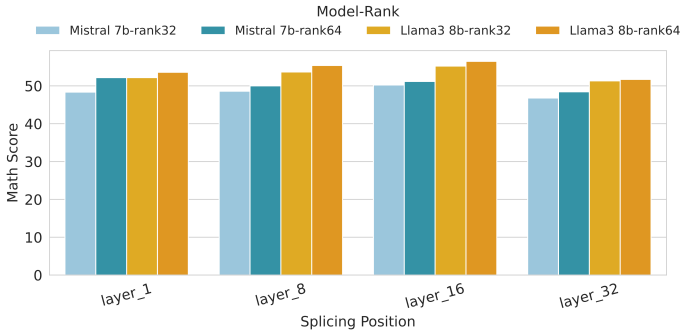

Fig. 2. Comparative performance analysis of distinct high-rank spliced layers within the SLoRA method on mathematical reasoning tasks.

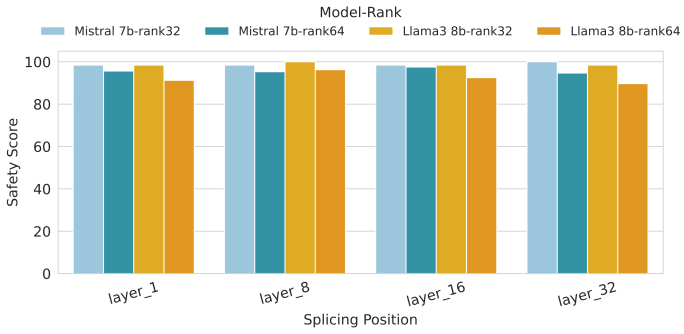

Fig. 3. Performance comparison of different high-rank splicing layers on safety alignment within the SLoRA method.

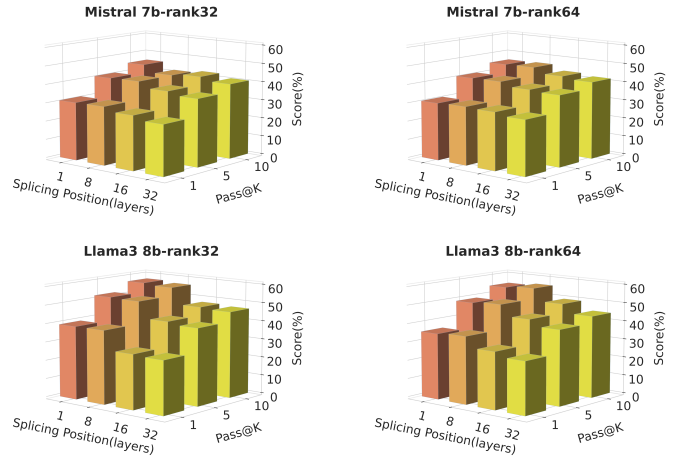

Fig. 4. Comparative analysis of code generation performance across distinct high-rank splicing layers within the SLoRA framework.

2) High-Rank Splice Position Analysis: To evaluate the hierarchical impact of high-rank splicing in SLoRA, we fixed the splicing factor at 4 and conducted experiments across mathematics, code, and safety datasets. Results in Figures 2 through 4 demonstrate that shifting the splice position from lower to intermediate layers yields performance gains in mathematics and code tasks, followed by a sharp decline at the final layer. This variation aligns with the hierarchical mechanism where Transformer models encode complex semantics primarily in deeper layers [24]. Specifically, introducing high-rank splicing at the topmost layer triggers feature distribution shifts that destabilize semantic consistency. Consequently, maintaining low-rank structures in the deepest layers preserves the balance between parameter efficiency and model integrity. The fluctuations observed in safety alignment scores at the highest layer further validate this layer-specific sensitivity.

3) Spliced Factor Analysis: Building on the layer position analysis, we set the start layer for high-rank splice to 8 to examine the impact of the splice factor parameter.

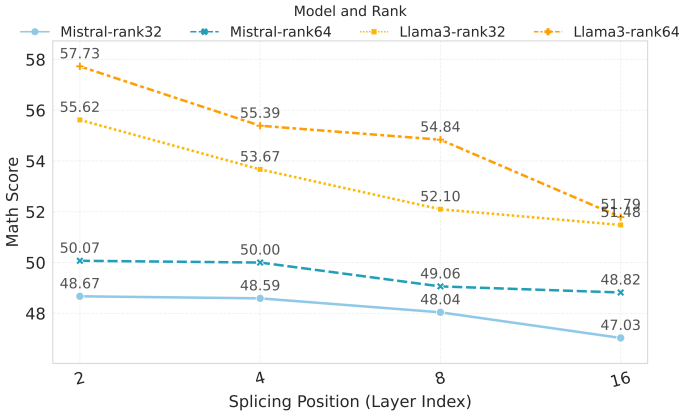

Fig. 5. Performance comparison of various splice factors within the SLoRA framework for mathematical reasoning tasks.

TABLE VII
COMPARATIVE ANALYSIS OF SPLICE FACTORS IN CODE GENERATION

Model-Rank	Splicing Factor	Pass@1	Pass@5	Pass@10
Llama-32	2	**41.46**	**54.47**	**59.67**
	4	39.63	51.82	56.35
	8	39.54	52.42	58.12
	16	38.23	51.42	57.29
Mistral-32	2	**32.92**	**43.71**	**47.36**
	4	31.52	41.43	41.43
	8	31.00	41.04	45.12
	16	31.15	40.97	45.24
Llama-64	2	35.15	46.99	51.51
	4	36.49	50.14	55.92
	8	37.07	51.13	57.81
	16	**37.37**	**52.84**	**58.83**
Mistral-64	2	**31.25**	40.71	44.43
	4	31.46	**41.30**	**45.80**
	8	30.97	41.20	44.59
	16	30.97	40.99	45.51

The experimental results presented in Figure 5 and Table VII demonstrate that performance on mathematical reasoning tasks is negatively correlated with the splice factor. This decline stems from the reduction in shared effective rank, suggesting that complex logical tasks necessitate the high representational capacity preserved by lower splice factors. Conversely, code generation exhibits varying sensitivity across configurations. While performance degrades with larger splice factors at a rank of 32, the rank 64 configuration maintains stability. This distinction implies that higher base ranks provide sufficient redundancy to mitigate the compression effects induced by the splicing mechanism.

V. Conclusion

SLoRA employs global shared weights to significantly reduce parameter counts while maintaining fine-tuning performance comparable to LoRA. We establish recommended parameter configurations based on a sensitivity analysis of high-rank splicing and splicing factors.

Future research will focus on achieving granular control over weight splicing and investigating fully shared strategies. Furthermore, reducing reliance on manual tuning through adaptive or automated hyperparameter optimization across diverse model architectures and tasks remains a critical direction for the evolution of SLoRA.

References

[1] T. B. Brown *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

[2] A. Chowdhery *et al.*, "Palm: Scaling language modeling with pathways," *J. Mach. Learn. Res.*, vol. 24, pp. 240:1–240:113, 2023. [Online]. Available: https://jmlr.org/papers/v24/22-1144.html

[3] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. [Online]. Available: https://openreview.net/forum?id=nZeVKeeFYf9

[4] D. J. Kopiczko, T. Blankevoort, and Y. M. Asano, "Vera: Vector-based random matrix adaptation," in *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. [Online]. Available: https://openreview.net/forum?id=NjNfLdxr3A

[5] Y. Li, S. Han, and S. Ji, "Vb-lora: Extreme parameter efficient fine-tuning with vector banks," in *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, and C. Zhang, Eds., 2024. [Online]. Available: http://papers.nips.cc/paper_files/paper/2024/hash/1e0d38c676d5855bcfab7f6d29d20ad9-Abstract-Conference.html

[6] L. Zhang, L. Zhang, S. Shi, X. Chu, and B. Li, "Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning," *CoRR*, vol. abs/2308.03303, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2308.03303

[7] S. Liu, C. Wang, H. Yin, P. Molchanov, Y. F. Wang, K. Cheng, and M. Chen, "Dora: Weight-decomposed low-rank adaptation," in *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. [Online]. Available: https://openreview.net/forum?id=3d5CIRG1n2

[8] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, S. Paul, B. Bossan, and M. Tietz, "PEFT: State-of-the-art parameter-efficient fine-tuning methods," https://github.com/huggingface/peft, 2022.

[9] K. Cobbe *et al.*, "Training verifiers to solve math word problems," *CoRR*, vol. abs/2110.14168, 2021. [Online]. Available: https://arxiv.org/abs/2110.14168

[10] C. Clark, K. Lee, M. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova, "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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 2924–2936. [Online]. Available: https://doi.org/10.18653/v1/n19-1300

[11] Y. Bisk, R. Zellers, J. Gao, Y. Choi *et al.*, "Piqa: Reasoning about physical commonsense in natural language," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 05, 2020, pp. 7432–7439.

[12] M. Sap, H. Rashkin, D. Chen, R. LeBras, and Y. Choi, "Socialiqa: Commonsense reasoning about social interactions," *arXiv preprint arXiv:1904.09728*, 2019.

[13] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord, "Think you have solved question answering? try arc, the ai2 reasoning challenge," *arXiv preprint arXiv:1803.05457*, 2018.

[14] T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal, "Can a suit of armor conduct electricity? A new dataset for open book question answering," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, Eds. Association for Computational Linguistics, 2018, pp. 2381–2391. [Online]. Available: https://doi.org/10.18653/v1/d18-1260

[15] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, "Hellaswag: Can a machine really finish your sentence?" in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, A. Korhonen, D. R. Traum, and L. Màrquez, Eds. Association for Computational Linguistics, 2019, pp. 4791–4800. [Online]. Available: https://doi.org/10.18653/v1/p19-1472

[16] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi, "Winogrande: an adversarial winograd schema challenge at scale," *Commun. ACM*, vol. 64, no. 9, pp. 99–106, 2021. [Online]. Available: https://doi.org/10.1145/3474381

[17] S. Chaudhary, "Code alpaca: An instruction-following llama model for code generation," 2023.

[18] M. Chen, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[19] F. Bianchi, M. Suzgun, G. Attanasio, P. Röttger, D. Jurafsky, T. Hashimoto, and J. Zou, "Safety-tuned llamas: Lessons from improving the safety of large language models that follow instructions," in *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. [Online]. Available: https://openreview.net/forum?id=gT5hALch9z

[20] X. Qi, Y. Zeng, T. Xie, P. Chen, R. Jia, P. Mittal, and P. Henderson, "Fine-tuning aligned language models compromises safety, even when users do not intend to!" in *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. [Online]. Available: https://openreview.net/forum?id=hTEGyKf0dZ

[21] A. Dubey *et al.*, "The llama 3 herd of models," *arXiv e-prints*, pp. arXiv–2407, 2024.

[22] M. Team, "Mistral 7b," *ArXiv, abs/2310.06825*, vol. 6, 2023.

[23] J. Zhang, J. You, A. Panda, and T. Goldstein, "Lori: Reducing cross-task interference in multi-task low-rank adaptation," *arXiv preprint arXiv:2504.07448*, 2025.

[24] J. Hewitt and C. D. Manning, "A structural probe for finding syntax in word representations," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 4129–4138. [Online]. Available: https://doi.org/10.18653/v1/n19-1419