RAC-LoRA: Router-Aware Combinatorial LoRA for Adaptive Parameter-Efficient Fine-Tuning

Anonymous authors
Paper under double-blind review

Abstract

Low-Rank Adaptation (LoRA) provides an efficient approach to parameter-efficient fine-tuning, while recent MoE-LoRA methods introduce multiple LoRA experts and input-dependent routing to improve adaptation capacity. However, existing designs commonly impose fixed relationships between the input-side and output-side low-rank factors, and the router mainly acts as an external selector rather than directly participating in expert computation. These constraints limit the flexibility of expert composition and weaken the coupling between routing and expert specialization. We propose RAC-LoRA (Router-Aware Combinatorial LoRA), which constructs $N_A N_B$ implicit experts by independently parameterizing A and B factors and dynamically routing among their combinations. In addition to expert selection, each router defines an input direction that conditions the corresponding expert while retaining the original representation. This enables the router to participate directly in expert computation without excessive information loss. We aggregate selected experts additively to allow complementary adaptation components to be jointly incorporated. Experiments on Llama-3.2-3B and Llama-3.1-8B show that RAC-LoRA achieves the highest average accuracy in the single-domain setting and the highest multi-task accuracy on both backbones among all compared methods.

1 Introduction

Parameter-efficient fine-tuning (PEFT) has become an effective paradigm for adapting large language models (LLMs) while substantially reducing the number of trainable parameters. Among existing approaches, Low-Rank Adaptation (LoRA) (Hu et al., 2022) represents the weight update as the product of two low-rank matrices, providing a simple and efficient alternative to full fine-tuning. However, a single LoRA adapter must accommodate heterogeneous patterns within the training data, which can lead to interference between distinct adaptation behaviors.

Motivated by the success of Mixture-of-Experts (MoE), recent studies have therefore introduced multiple LoRA experts and a router to provide input-dependent adaptation, including LoRAMoE (Dou et al., 2024), HydraLoRA (Tian et al., 2024), and subsequent collaborative LoRA variants (Zhou et al., 2025). These methods demonstrate the potential of combining MoE-style routing with low-rank adaptation, but the interaction between the routing mechanism and the internal structure of LoRA experts remains largely constrained.

Despite this progress, existing MoE-LoRA designs face two structural limitations. First, the A and B factors of an expert are typically coupled through a predetermined relationship. Such fixed pairing implicitly assumes that a particular input-side transformation should always be associated with a particular output-side transformation. However, the most suitable input and output adaptations may vary across samples, and this assumption can unnecessarily restrict the space of realizable low-rank transformations.

Second, routing and expert computation are often loosely coupled. In a conventional MoE-LoRA formulation, the router primarily determines which independently parameterized LoRA branches contribute to the output. The selected expert then processes the original input without explicitly incorporating the information encoded by the router. This separation limits the role of the router

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

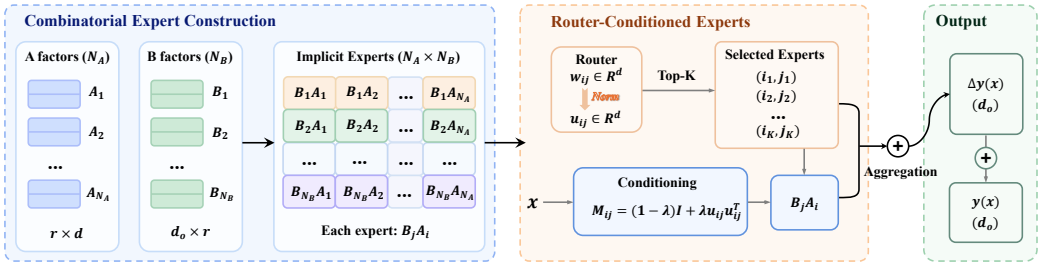


Figure 1: Architecture of RAC-LoRA.

from an architectural selector to an external gating mechanism, rather than using it as a source of task-specific information for expert specialization.

To address these limitations, we propose RAC-LoRA (Router-Aware Combinatorial LoRA), a unified architecture that couples expert construction, routing, and input transformation, as illustrated in Figure 1. Instead of treating each LoRA branch as a fixed A–B pair, RAC-LoRA maintains a set of input-side factors $\{A_i\}_{i=1}^{N_A}$ and output-side factors $\{B_j\}_{j=1}^{N_B}$, from which $N_A N_B$ implicit experts $B_j A_i$ are formed. A router dynamically selects a subset of these combinations for each input, allowing the input and output sides to specialize independently and adaptively. Beyond expert selection, each router vector also defines an input direction associated with its expert. We use this direction to condition the expert input while retaining the original representation, thereby coupling the router with the computation performed by the selected expert. Finally, the selected experts are directly aggregated, allowing different experts to contribute complementary adaptation components rather than forcing them into a probability-normalized mixture.

Our contributions are summarized as follows:

- We propose RAC-LoRA, a router-aware combinatorial LoRA architecture that constructs $N_A N_B$ implicit experts from independently parameterized A and B factors. Hence the fixed pairing constraint has been removed between input-side and output-side adaptations. We further introduce router-conditioned expert computation, where each router not only selects experts but also conditions their input representations, establishing a direct interaction between routing decisions and LoRA expert specialization.
- We provide a unified theoretical analysis of RAC-LoRA, showing the advantage of the combinatorial representation space and deriving the router-conditioned transformation from an explicit optimization objective with a controllable information distortion.
- We conduct extensive experiments across multiple models and downstream tasks to evaluate the effectiveness of RAC-LoRA and analyze the contribution of its individual components.

2 Related Work

2.1 LoRA and Factorized Adaptation

Low-Rank Adaptation (LoRA) (Hu et al., 2022) parameterizes the weight update $\Delta W \in R^{d_o \times d}$ of a pretrained linear layer as a low-rank product,

$$\Delta W = BA, \tag{1}$$

where $A \in \mathbb{R}^{r \times d}$ and $B \in \mathbb{R}^{d_o \times r}$ with $r \ll \min(d, d_o)$. By freezing the backbone parameters and optimizing only the low-rank factors, LoRA substantially reduces the trainable parameter count while retaining strong adaptation capacity.

A number of subsequent studies improve the expressiveness or optimization of this factorization. AdaLoRA (Zhang et al., 2023) adaptively allocates the rank budget across layers, while SaLoRA and MoSLoRA (Hu et al., 2023; Wu et al., 2024) explore alternative parameterizations and aggregation mechanisms. DoRA (Liu et al., 2024) decomposes the update into magnitude and directional components, while PiSSA (Meng et al., 2024) initializes the low-rank factors from principal singular

components of the pretrained weights. Other works investigate the optimization dynamics of LoRA from gradient or update perspectives, including LoRA-GA, LoRA-Pro, and related gradient-aware methods (Wang et al., 2024; 2025). These studies primarily focus on improving the factorization or optimization of individual LoRA modules. In contrast, our work investigates how multiple low-rank factors can be structurally organized and adaptively combined to form input-dependent experts.

2.2 Mixture-of-Experts and Adaptive Routing

Mixture-of-Experts (MoE) introduces conditional computation by routing each input to a subset of specialized experts (Shazeer et al., 2017). Sparse routing mechanisms, such as those used in Switch Transformer and GShard (Fedus et al., 2022; Lepikhin et al., 2021), reduce computation by activating only a small number of experts for each token. Subsequent studies have explored more stable and expressive routing through load balancing, auxiliary objectives, and alternative parameterizations.

Beyond conventional learned routers, recent work has investigated tighter interactions between routing and expert representations. Dense routing approaches (Panda et al., 2025) provide broader gradient signals during training, while ERC (Lv et al., 2026) explicitly associates routing representations with individual experts. AoE (Lv et al., 2025) derives routing behavior from expert parameters, and CompeteSMoE (Pham et al., 2024) determines expert contributions from their activation behavior. These studies suggest that the relationship between the router and the experts is itself an important design dimension. Nevertheless, conventional MoE architectures typically assume that each expert is an independently parameterized function.

2.3 MoE-LoRA and Expert Architecture

Combining LoRA with MoE provides a parameter-efficient way to introduce specialization into pretrained language models. LoRAMoE (Dou et al., 2024) introduces multiple LoRA adapters and employs a router to determine their contributions, demonstrating that expert specialization can alleviate the limitations of a single LoRA module. MixLoRA (Chen et al., 2025) incorporates multiple LoRA-based experts into transformer blocks with top-k routing and auxiliary load balancing. MoLA (Gao et al., 2025) and AlphaLoRA (Qing et al., 2024) further study how the number and allocation of LoRA experts should vary across layers, motivated by the observation that uniform expert allocation can lead to redundancy.

A complementary line of work changes the structure of the low-rank factors themselves. HydraLoRA (Tian et al., 2024) adopts an asymmetric design with a shared A matrix and multiple B matrices. R-LoRA (Liu et al., 2025) further targets the similarity of multi-head LoRA representations through random initialization and multi-head dropout. More recently, CoLA (Zhou et al., 2025) explores several collaborative strategies for exploiting their relationships. These approaches motivate a broader view of LoRA experts beyond conventional one-to-one factorization. However, they primarily modify the factor-sharing or factor-collaboration structure, while the possible combinations between independently parameterized A and B factors are not explicitly treated as a jointly routed expert space.

Another recent direction is to make the MoE mechanism implicit rather than explicitly maintaining a conventional router. FlyLoRA (Zou et al., 2025) performs rank-wise expert activation and uses a frozen sparse random projection to couple routing with the down-projection. This line of work demonstrates that routing and low-rank computation can be closely integrated, but its specialization is organized at the rank level rather than through combinations of independent input-side and output-side factors.

3 Methodology

We propose **RAC-LoRA (Router-Aware Combinatorial LoRA)**, which addresses two limitations of existing MoE-LoRA architectures: the rigid pairing between the input-side and output-side low-rank factors, and the weak interaction between routing and expert computation. RAC-LoRA constructs a combinatorial set of implicit experts from independently parameterized A and B factors, and each router further conditions the representation processed by its corresponding expert. The re-

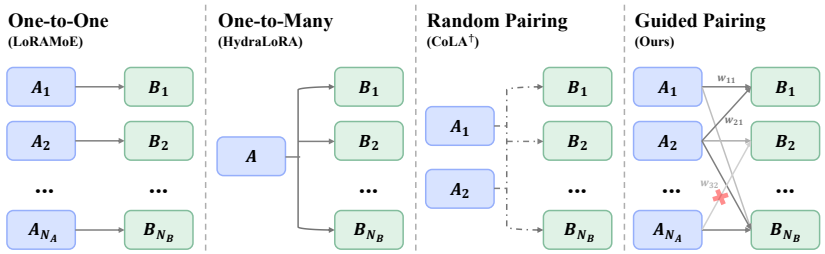


Figure 2: Comparison of LoRA factor-combination paradigms. RAC-LoRA assigns a router w_{ij} to each candidate A/B pair and performs input-dependent Top-K selection. Crosses indicate unselected pairs.

sulting architecture provides flexible input-output compositions while coupling expert specialization with routing decisions.

3.1 Combinatorial A/B Experts

Consider a linear layer with input dimension d and output dimension d_o. RAC-LoRA introduces N_A input-side factors $A_i \in \mathbb{R}^{r\times d}$ and N_B output-side factors $B_j \in \mathbb{R}^{d_o\times r}$. Each pair (A_i, B_j) forms an implicit expert

$$E_{ij}(x) = B_j A_i x. \tag{2}$$

Each implicit expert is associated with a router vector $w_{ij} \in \mathbb{R}^d$. Given an input token x, its routing score is

$$s_{ij}(x) = w_{ij}^\top x. \tag{3}$$

The router selects the K experts with the largest scores, denoted by $\mathcal{S}_K(x) = \mathrm{TopK}(s_{ij}(x))$. During the forward pass, the selected experts are directly added rather than normalized into a probability-weighted mixture. We implement the corresponding routing operation with a straight-through estimator, using the binary Top-K mask in the forward pass and a masked softmax as the surrogate gradient in the backward pass. Figure 2 compares experts combination of RAC-LoRA with other forms in previous works.

Representation-space analysis. The combinatorial construction removes the structural restriction imposed by fixed A/B pairings. Let $\mathcal{E}_f$ denote the set of available A/B pairs in a fixed-pairing architecture and let $\mathcal{E}_c = \{(i,j) \mid 1 \le i \le N_A,\ 1 \le j \le N_B\}$ denote the complete combinatorial set. By construction, $\mathcal{E}_f \subseteq \mathcal{E}_c$. Define the corresponding linear operator spaces as

$$\mathcal{V}_f = \mathrm{span}\{B_j A_i : (i,j) \in \mathcal{E}_f\}, \qquad \mathcal{V}_c = \mathrm{span}\{B_j A_i : (i,j) \in \mathcal{E}_c\}. \tag{4}$$

Therefore,

$$\mathcal{V}_f \subseteq \mathcal{V}_c. \tag{5}$$

For a target update matrix T, let $\varepsilon(T;\mathcal{V}) = \min_{M\in\mathcal{V}} \|T - M\|_F$ denote its best approximation error in $\mathcal{V}$. Since $\mathcal{V}_f \subseteq \mathcal{V}_c$, their best approximation errors satisfy

$$\varepsilon(T;\mathcal{V}_c) \le \varepsilon(T;\mathcal{V}_f). \tag{6}$$

Hence, at the level of the linear-span relaxation, the expanded candidate dictionary admits no larger best approximation error than fixed pairing. In the hard-routing model, cross-pair transformations such as B_3A_1 remain available for input-dependent Top-K selection, allowing input- and output-side adaptations to be selected independently.

3.2 Router-Conditioned Experts

The combinatorial construction enlarges the candidate expert space, but routing alone does not directly modify expert computation. RAC-LoRA therefore uses each router vector not only to select an expert, but also to condition the input representation processed by that expert.

For each expert (i, j), we define the unit router direction $u_{ij} = w_{ij}/\|w_{ij}\|_2$ and the corresponding rank-one orthogonal projector $P_{ij} = u_{ij}u_{ij}^\top$. RAC-LoRA then transforms the input using

$$M_{ij} = (1-\lambda)I + \lambda P_{ij}, \qquad 0 \le \lambda < 1. \tag{7}$$

The conditioned expert output is

$$z_{ij}(x) = B_j A_i M_{ij} x. \tag{8}$$

Let α denotes LoRA alpha, the update can be written as

$$\Delta y(x) = \frac{\alpha}{r} \sum_{(i,j)\in\mathcal{S}_K(x)} B_j A_i M_{ij} x. \tag{9}$$

Router–expert coupling. The routing mechanism in conventional MoE-LoRA only determines the contribution of each expert, while the computation performed by the expert is independent of the router. Let $\mathbf{w} = \{w_{ij}\}_{i,j}$ denote the collection of router vectors, and let $h_{ij}(\mathbf{w}, x)$ denote the routing coefficient obtained after Top-K selection and the corresponding masked normalization. A conventional MoE-LoRA layer can be abstractly written as

$$F(x) = \sum_{i,j} h_{ij}(\mathbf{w}, x) E_{ij}(x) = \sum_{i,j} h_{ij}(\mathbf{w}, x) B_j A_i x, \tag{10}$$

where $E_{ij}(x)$ is independent of the router parameters. In RAC-LoRA, the same router vector w_{ij} additionally determines the input transformation $M_{ij}(w_{ij})$ defined in Eq. 7, yielding

$$F(x) = \sum_{i,j} h_{ij}(\mathbf{w}, x) B_j A_i M_{ij}(w_{ij}) x. \tag{11}$$

During backpropagation, the hard Top-K routing coefficients are optimized using a masked-softmax straight-through estimator. The resulting gradient with respect to w_{ij} contains two paths:

$$\frac{\partial F}{\partial w_{ij}} = \sum_{k,l} \frac{\partial h_{kl}}{\partial w_{ij}} B_l A_k M_{kl} x + h_{ij} B_j A_i \frac{\partial M_{ij}}{\partial w_{ij}} x. \tag{12}$$

The first term captures the effect of changing routing preferences, while the second term directly changes the computation performed by the selected expert. Since M_{ij} depends on w_{ij}, the second term is generally nonzero. Thus, RAC-LoRA couples routing with expert computation. The router not only determines *which* expert is activated, but also influences *how* the selected expert processes the input.

Optimization view of router conditioning. The form of M_{ij} can be derived from a simple regularized representation objective. For a fixed router direction u and $P = uu^\top$, consider

$$z^\star = \arg\min_z \frac{1}{2}\|z - x\|_2^2 + \frac{\mu}{2}\|(I-P)z\|_2^2, \mu \ge 0. \tag{13}$$

The first term preserves the original representation, while the second penalizes components orthogonal to the router direction. Decomposing $x = Px + (I-P)x$ and $z = Pz + (I-P)z$, and using the orthogonality of P and $I - P$, the objective separates into the parallel and orthogonal subspaces. Its unique minimizer satisfies

$$Pz^\star = Px, (I-P)z^\star = \frac{1}{1+\mu}(I-P)x. \tag{14}$$

Hence,

$$z^\star = Px + \frac{1}{1+\mu}(I-P)x. \tag{15}$$

Setting $1 - \lambda = (1+\mu)^{-1}$, equivalently $\mu = \lambda/(1-\lambda)$, gives

$$z^\star = [(1-\lambda)I + \lambda uu^\top]x. \tag{16}$$

Therefore, the proposed router-conditioned representation is the closed-form optimum of an objective that preserves the original input while selectively suppressing components orthogonal to the router direction. The transformation is consequently not an arbitrary interpolation between x and its projection.

Information distortion Inequality. The necessity of retaining the original input can be quantified directly. Under pure projection, the transformed input is Px, and the discarded component is $(I - P)x$, yielding

$$\|x - Px\|_2 = \|(I - P)x\|_2. \tag{17}$$

Because P is a rank-one projector, there exists a unit-norm x orthogonal to u for which $Px = 0$, and therefore

$$\sup_{\|x\|_2=1} \|x - Px\|_2 = 1. \tag{18}$$

Thus, a pure router projection can completely remove the input for directions orthogonal to the router.

In contrast, RAC-LoRA applies $M = (1 - \lambda)I + \lambda P$, for which

$$x - Mx = \lambda(I - P)x. \tag{19}$$

Consequently,

$$\|x - Mx\|_2 = \lambda\|x - Px\|_2. \tag{20}$$

The induced distortion is therefore reduced to a factor λ of that caused by pure projection. Moreover, M has eigenvalue 1 along u and eigenvalue $1 - \lambda$ on the orthogonal complement of u. Hence,

$$(1 - \lambda)\|x\|_2 \leq \|Mx\|_2 \leq \|x\|_2, \tag{21}$$

and M remains full-rank for $\lambda < 1$. These results show that RAC-LoRA emphasizes the router-selected direction while retaining all input dimensions and controlling the magnitude of the induced distortion.

4 Experiments

4.1 Experimental Setup

Models and baselines. We evaluate RAC-LoRA on Llama-3.2-3B and Llama-3.1-8B. We compare against standard LoRA and three representative MoE-LoRA methods, including LoRAMoE, HydraLoRA, and CoLA. For RAC-LoRA, we use $N_A = N_B = 3$, yielding 9 implicit experts, with Top-$K = 6$, $\lambda = 0.5$, and $\alpha = 32$. All compared methods target the same backbone modules and are trained under the same optimization protocol. Experiments are conducted on NVIDIA A800 GPUs with 80GB memory. Each configuration is independently trained with three random seeds, and we report the mean and standard deviation across the three runs. Full training, routing, and evaluation configurations for all methods are provided in Appendix C.

Evaluation settings. We consider single-domain and multi-task adaptation, evaluated with LlamaFactory (Zheng et al., 2024). The single-domain setting contains five training–evaluation pairs covering general instruction following, legal knowledge, medical knowledge, mathematical reasoning, and general language understanding. The multi-task setting uses OpenOrca for training and BBH for evaluation. The complete benchmark configurations are shown in Table 1. Accuracy is used as the evaluation metric throughout the experiments.

Table 1: Training and evaluation datasets used in our experiments.

Setting	Training Data	Evaluation Data
Single-domain	Databricks-Dolly-15k (Conover et al., 2023)	MMLU (Hendrycks et al., 2021)
Single-domain	US-Terms (Chalkidis et al., 2023) + Lawyer-Instruct (Alignment-Lab-AI, 2024)	MMLU-Law
Single-domain	iCliniq + GenMedGPT-5k (Li et al., 2023)	MMLU-Medicine
Single-domain	GSM8K (Cobbe et al., 2021)	GSM8K-Test
Single-domain	FineVal-EN (Wang et al., 2023)	FineVal-Test
Multi-task	OpenOrca (Lian et al., 2023)	BBH (Suzgun et al., 2023)

Table 2: Single-domain adaptation results on Llama-3.2-3B and Llama-3.1-8B. Accuracy is reported as mean ± standard deviation over three random seeds and best results are **bolded**.

Adapters	LoRA	LoRAMoE	HydraLoRA	CoLA$^{\intercal}$	RAC-LoRA
A / B	1 / 1	4 / 4	1 / 6	4 / 4	3 / 3
Llama-3.2-3B					
MMLU-Overall	34.53 ± 0.80	29.01 ± 1.59	28.64 ± 1.53	40.89 ± 0.96	**42.97 ± 0.75**
MMLU-Law	24.60 ± 0.23	24.50 ± 0.47	24.35 ± 0.29	24.52 ± 0.15	**24.68 ± 0.16**
MMLU-Medicine	27.19 ± 0.95	26.01 ± 0.31	25.85 ± 0.28	27.21 ± 0.83	**30.76 ± 0.75**
GSM8K	51.75 ± 0.31	48.04 ± 1.81	46.09 ± 1.99	51.40 ± 1.21	**52.21 ± 0.74**
FineVal	38.49 ± 2.41	36.73 ± 0.35	35.85 ± 1.41	**38.75 ± 0.71**	38.24 ± 0.71
Average	35.31 ± 0.41	32.86 ± 0.78	32.16 ± 0.78	36.55 ± 0.46	**37.77 ± 0.42**
Llama-3.1-8B					
MMLU-Overall	48.64 ± 4.49	49.01 ± 1.70	42.06 ± 0.86	55.98 ± 1.71	**59.57 ± 0.17**
MMLU-Law	25.79 ± 0.16	25.35 ± 0.44	25.39 ± 0.39	26.85 ± 1.31	**28.74 ± 1.81**
MMLU-Medicine	42.18 ± 2.04	40.50 ± 2.34	35.13 ± 3.08	41.91 ± 4.33	**43.16 ± 3.60**
GSM8K	56.18 ± 0.22	53.30 ± 0.59	52.64 ± 0.41	55.67 ± 1.09	**56.28 ± 0.60**
FineVal	47.67 ± 2.19	44.78 ± 1.17	44.02 ± 0.99	48.81 ± 2.76	**49.81 ± 2.46**
Average	44.09 ± 1.08	42.59 ± 0.04	39.85 ± 0.35	45.84 ± 0.91	**47.51 ± 0.41**

Table 3: Multi-task adaptation results on Llama-3.2-3B and Llama-3.1-8B.

Task	LoRA	LoRAMoE	HydraLoRA	CoLA$^{\intercal}$	RAC-LoRA
Llama-3.2-3B	36.13 ± 0.21	34.63 ± 0.46	33.91 ± 0.71	36.18 ± 0.19	**36.79 ± 0.21**
Llama-3.1-8B	42.02 ± 0.43	41.69 ± 0.14	41.49 ± 0.17	42.55 ± 0.29	**43.04 ± 0.36**

4.2 Comparison with Existing Methods

We first compare RAC-LoRA with LoRA, LoRAMoE, HydraLoRA, and CoLA on both Llama-3.2-3B and Llama-3.1-8B. This comparison evaluates whether the proposed architecture provides consistent improvements across model scales and different adaptation settings.

4.2.1 Single-Domain Adaptation

Table 2 reports the results on Llama-3.2-3B and Llama-3.1-8B. RAC-LoRA consistently improves over standard LoRA and the existing MoE-LoRA baselines across the five single-domain settings. The gains are particularly evident on the domain-specific benchmarks, where the training and evaluation distributions emphasize specialized knowledge. Across both model scales, RAC-LoRA achieves consistent improvements over the compared methods. The improvements are not confined to a particular evaluation domain, suggesting that the proposed combination of flexible A/B composition and router-conditioned computation provides a general mechanism for input-dependent adaptation.

4.2.2 Multi-Task Adaptation

We next evaluate the methods in the multi-task setting, where OpenOrca provides heterogeneous instruction-following data and BBH evaluates a diverse collection of reasoning tasks. Table 3 reports the results for both model scales. RAC-LoRA obtains the strongest performance on both model scales. The consistent improvement in the multi-task setting indicates that the proposed architecture can exploit multiple adaptation patterns within a single model, rather than being limited to a particular single-domain specialization.

4.3 Ablation Study

We conduct all ablation experiments on the Databricks-Dolly-15k and MMLU benchmark using Llama-3.1-8B. Each ablation isolates one component of RAC-LoRA while keeping the remaining

Table 4: Ablation studies on Llama-3.1-8B.

Variants	MMLU Accuracy (%)	Δ Accuracy (pp)
Pairing Strategy		
Fixed Pairing	57.85 ± 0.31	-1.72
Random Pairing	57.71 ± 1.51	-1.86
Combinatorial Routing	$\mathbf{59.57 \pm 0.17}$	**0.0**
Conditioning Strategy		
No Conditioning	58.41 ± 0.70	-1.16
Pure Projection	51.38 ± 0.75	-8.17
Residual Conditioning	$\mathbf{59.57 \pm 0.17}$	**0.0**
Aggregation		
Masked Softmax	51.68 ± 0.74	-7.89
Hard Mean	52.11 ± 0.93	-7.46
Hard Sum	$\mathbf{59.57 \pm 0.17}$	**0.0**

Table 5: Effect of the conditioning coefficient λ on Llama-3.1-8B.

λ	0	0.25	0.5	0.75	0.9
Llama-3.1-8B	58.41 ± 0.70	59.23 ± 0.28	$\mathbf{59.57 \pm 0.17}$	57.75 ± 0.28	53.21 ± 0.28

architecture and training protocol unchanged. We summarize the ablation results in Table 4 and report the corresponding MMLU accuracy.

Ablation of A/B Pairing. We first investigate whether the combinatorial A/B construction contributes to the observed improvements. We compare three pairing strategies while keeping the number and dimensions of the trainable factors fixed: (i) fixed pairing, where each A_i is permanently associated with two B_j; (ii) random fixed pairing, where the A/B associations are randomly initialized and then kept fixed; and (iii) combinatorial routing, where all $N_A N_B$ combinations are available to the router.

The combinatorial variant provides a clear improvement over both fixed alternatives. Since the number of trainable factors is unchanged, the result indicates that the gain is attributable to the flexible composition of A/B factors rather than simply to increasing parameter capacity. This observation is consistent with the representation-space analysis in Section 3.1.

Ablation of Router Conditioning. We next examine whether the router should directly participate in expert computation. We compare three variants: *No Conditioning*, where selected experts compute $B_j A_i x$; *Pure Projection*, where the input is replaced by its router projection $P_{ij} x$; and *Residual Conditioning*, which uses the proposed transformation $M_{ij} = (1-\lambda)I + \lambda P_{ij}$. Removing conditioning results in a performance drop, indicating that the router provides useful information beyond expert selection. Pure projection performs worse than the residual formulation, consistent with the information loss associated with replacing the original representation by a rank-one projection. The proposed residual conditioning achieves the strongest result by emphasizing the router-selected direction while retaining the original input.

Ablation of Expert Aggregation. Finally, we investigate how the selected experts should be aggregated. We compare the proposed hard summation with a masked softmax mixture and a hard mean. The same Top-K experts are selected in all three variants. Under the default adapter scaling, hard summation achieves the highest accuracy among the evaluated aggregation rules.

4.4 Analysis of Routing Hyperparameters

We further analyze the sensitivity of RAC-LoRA to the router-conditioning coefficient λ and the number of active experts K. All experiments in this section use Llama-3.1-8B on Databricks-Dolly-15k and MMLU, with all other settings fixed to the default configuration.

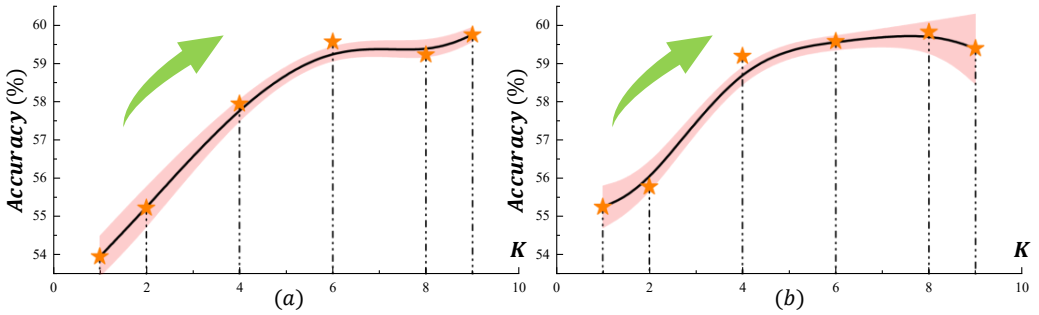


Figure 3: Effect of the number of selected experts K on Llama-3.1-8B (a) without and (b) with scale control ($c_K = \frac{6}{K}$). Lines show the mean over three seeds and shaded regions show $\pm$ std. Under scale control, accuracy still increases with K up to 8.

4.4.1 Effect of λ

The coefficient λ controls the strength of router-conditioned input transformation. When $\lambda = 0$, router conditioning is removed, whereas larger values place greater emphasis on the router-defined direction. The performance exhibits a non-monotonic dependence on λ. Moderate conditioning provides the strongest result, while overly aggressive conditioning reduces performance. This trend is consistent with the theoretical formulation in Section 3.2: increasing λ strengthens the router-induced directional bias while simultaneously increasing the deviation from the original representation. We therefore use $\lambda = 0.5$ as the default value.

4.4.2 Effect of Top-K

We vary K while keeping all other settings fixed (Figure 3(a)). Since RAC-LoRA directly sums the selected experts, increasing K raises both the number of active components and the magnitude of the aggregated update. To separate these effects, we conduct a scale-controlled sweep, rescaling the aggregated LoRA update by

$$c_K = \frac{6}{K}, \tag{22}$$

so that its contribution is normalized to the reference scale at $K = 6$ (Figure 3(b)). In both settings, accuracy improves from $K = 1$ to $K = 6$ and then plateaus; under scale control, it still increases from 55.25 to 59.57, showing that the gain is not merely due to update magnitude. We use $K = 6$, which attains the plateau with fewer active experts.

4.5 Additional Studies

Beyond the main ablation and routing analyses, we further investigate the architectural scalability and computational characteristics of RAC-LoRA. In particular, Appendix A studies three aspects that are not included in the main text due to space constraints: the effect of the N_A/N_B ratio, a linear-scaling router design for larger factor configurations, and a detailed computational efficiency comparison with existing MoE-LoRA methods. These experiments provide additional evidence on the scalability, practical overhead, and architectural flexibility of RAC-LoRA.

5 Conclusion

We presented RAC-LoRA, a router-aware combinatorial LoRA framework that decouples A/B factor pairing and directly integrates routing with expert computation. By combining flexible A/B composition with router-conditioned representations, RAC-LoRA provides a more adaptive and expressive MoE-LoRA architecture. Experiments on two Llama backbones demonstrate RAC-LoRA's effectiveness in both single-domain and multi-task adaptation, while focused component analyses examine the contributions of its core design choices. Building on these results, future work will extend RAC-LoRA to broader model families, larger model and data scales, and additional adaptation regimes.

AI USE STATEMENT

In this work, we used generative AI tools to revise and polish the manuscript and to assist with writing and debugging auxiliary code. We did not use generative AI tools to implement the core method, generate research ideas, develop the theory, or produce figures; other disclosure categories (e.g., dataset creation) are not applicable. All AI-assisted text and code were reviewed, tested, and verified by the authors, who take full responsibility for the content of this paper.

REFERENCES

Alignment-Lab-AI. Lawyer-instruct. https://huggingface.co/datasets/Alignment-Lab-AI/Lawyer-Instruct, 2024.

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

Ronghuai Chen, Ce Yu, Hao Fu, Xiaoteng Hu, and Bin Yang. MixLoRA: An efficient multi-tenant framework for concurrently serving diverse LoRA models in large language models. In *Proceedings of the 54th International Conference on Parallel Processing*, 2025.

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

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free dolly: Introducing the world's first truly open instruction-tuned LLM, 2023. URL https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. LoRAMoE: Alleviating world knowledge forgetting in large language models via MoE-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(1), 2022.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Ruibo Liu, Baochen Sun, Yawen Zhang, Daiyi Peng, Xiaoyuan Guo, and Vs Subrahmanian. MoLA: MoE LoRA with layer-wise expert allocation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, 2025.

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Yahao Hu, Yifei Xie, Tianfeng Wang, Man Chen, and Zhisong Pan. Structure-aware low-rank adaptation for parameter-efficient fine-tuning. *Mathematics*, 2023.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *9th International Conference on Learning Representations, ICLR 2021*, 2021.

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

Wing Lian, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". OpenORCA: An open dataset of GPT augmented flan reasoning traces. https://huggingface.co/datasets/Open-Orca/OpenOrca, 2023.

Jinda Liu, Yi Chang, and Yuan Wu. R-LoRA: Randomized multi-head LoRA for efficient multi-task learning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: weight-decomposed low-rank adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Ang Lv, Ruobing Xie, Yining Qian, Songhao Wu, Xingwu Sun, Zhanhui Kang, Di Wang, and Rui Yan. Autonomy-of-experts models. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Ang Lv, Jin Ma, Yiyuan Ma, and Siyuan Qiao. Coupling experts and routers in mixture-of-experts via an auxiliary loss. In *International Conference on Learning Representations*, 2026.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems*, 2024.

Ashwinee Panda, Vatsal Baherwani, Zain Sarwar, Benjamin Thérien, Sambit Sahu, Tom Goldstein, and Supriyo Chakraborty. Dense backpropagation improves training for sparse mixture-of-experts. In *Advances in Neural Information Processing Systems*, 2025.

Quang Pham, Giang Do, Huy Nguyen, TrungTin Nguyen, Chenghao Liu, Mina Sartipi, Binh T. Nguyen, Savitha Ramasamy, Xiaoli Li, Steven Hoi, and Nhat Ho. CompeteSMoE – effective training of sparse mixture of experts via competition. *arXiv preprint arXiv:2402.02526*, 2024.

Peijun Qing, Chongyang Gao, Yefan Zhou, Xingjian Diao, Yaoqing Yang, and Soroush Vosoughi. AlphaLoRA: Assigning LoRA experts based on layer training quality. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, 2023.

Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Chengzhong Xu. HydraLoRA: An asymmetric LoRA architecture for efficient fine-tuning. In *Advances in Neural Information Processing Systems*, 2024.

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

Shaowen Wang, Linxi Yu, and Jian Li. LoRA-GA: Low-rank adaptation with gradient approximation. In *Advances in Neural Information Processing Systems*, 2024.

Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. LoRA-Pro: Are low-rank adapters properly optimized? In *The Thirteenth International Conference on Learning Representations*, 2025.

Taiqiang Wu, Jiahao Wang, Zhe Zhao, and Ngai Wong. Mixture-of-subspaces in low-rank adaptation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.

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

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

Yiyun Zhou, Chang Yao, and Jingyuan Chen. CoLA: Collaborative low-rank adaptation. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.

Heming Zou, Yunliang Zang, Wutong Xu, Yao Zhu, and Xiangyang Ji. Flylora: Boosting task decoupling and parameter efficiency via implicit rank-wise mixture-of-experts. In *Advances in Neural Information Processing Systems*, 2025.

A Supplementary Experiments

A.1 A/B Ratio Analysis

RAC-LoRA does not require $N_A = N_B$. We therefore vary the numbers of input-side and output-side factors while keeping the overall training protocol unchanged. The number of implicit experts is determined by the product $N_A N_B$.

Table 6: Results with different A/B ratios on Llama-3.1-8B.

N_A / N_B	1 / 5	2 / 4	3 / 3	4 / 2	5 / 1
Llama-3.1-8B	58.11 ± 0.96	58.67 ± 0.57	$\underline{59.57 \pm 0.17}$	$\mathbf{59.81 \pm 0.42}$	59.06 ± 0.62

The results in Table 6 show that RAC-LoRA remains effective when N_A and N_B are unequal, demonstrating that the proposed architecture does not rely on symmetric factor counts. The 4×2 configuration obtains the highest mean accuracy, but its margin over the balanced 3×3 configuration is within one standard deviation; we therefore keep 3×3 as a symmetric default.

A.2 Linear-Scaling Router for Larger A/B Configurations

A.2.1 Linear-Scaling Router Design

The full combinatorial router produces $N_A N_B$ routing outputs. As the numbers of A and B factors increase, the router parameter count therefore grows quadratically when N_A and N_B are scaled together. Specifically, with input dimension d, the dense router contains

$$P_{\text{dense}} = N_A N_B d \tag{23}$$

trainable parameters. This quadratic growth can become a practical bottleneck even though the LoRA factor parameters themselves scale only linearly with $N_A + N_B$.

To address this issue, we introduce a locally constrained pairing strategy for the router. More generally, an arbitrary number of admissible pairings can be prescribed in advance, and the router is constrained to select only from these designated pairings. For the symmetric case $N_A = N_B$, we provide a construction as follows. Specifically, each A_i is allowed to interact with at most three neighboring B factors, and symmetrically each B_j is associated with at most three neighboring A factors. For example, an interior factor can use

$$A_i \leftrightarrow B_{i-1}, B_i, B_{i+1}, \tag{24}$$

with the corresponding boundary factors truncated to the valid index range. Thus, the dense set of all $N_A N_B$ combinations is replaced by a local set of candidate pairings whose cardinality grows linearly with the number of factors. Specifically, the router contains at most $P_{\text{local}} = 3N_A d$ trainable parameters, changing from quadratic growth in the dense combinatorial design to linear growth under the constrained design.

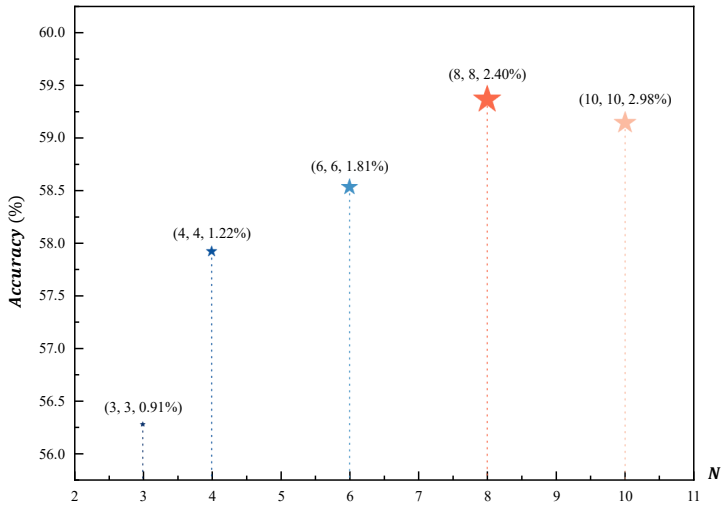


Figure 4: Scaling results on GSM8K. The x-axis denotes the numbers of N_A and N_B, and the y-axis reports GSM8K accuracy. Each point is annotated with the triple $(N_A, N_B, \text{params})$, where params denotes the number of trainable parameters.

A.2.2 Scaling Behavior

We study the scaling behavior of RAC-LoRA with a linear-scaling router on GSM8K with Llama-3.1-8B. The results are shown in Figure 4. We vary the number of factors under symmetric configurations, i.e., $N_A = N_B$, and measure the resulting task performance. The results show a clear scaling trend. As N_A and N_B increase, performance improves consistently, indicating that a larger number of locally constrained factor pairs enables the router to model richer interactions between the input- and output-side factors. This improvement is sustained across the tested range and eventually saturates at $N_A = N_B = 8$, beyond which no further gains are observed. These results demonstrate that our method exhibits favorable parameter scaling behavior, while also indicating that unbounded parameter expansion does not guarantee perpetual performance improvement.

A.3 Efficiency Analysis

We further compare the computational characteristics of RAC-LoRA with the evaluated MoE-LoRA baselines. All methods are trained under the same hardware and training protocol. We record average CPU utilization, peak GPU memory consumption, peak GPU reservation, and wall-clock training time, as shown in Figure 5.

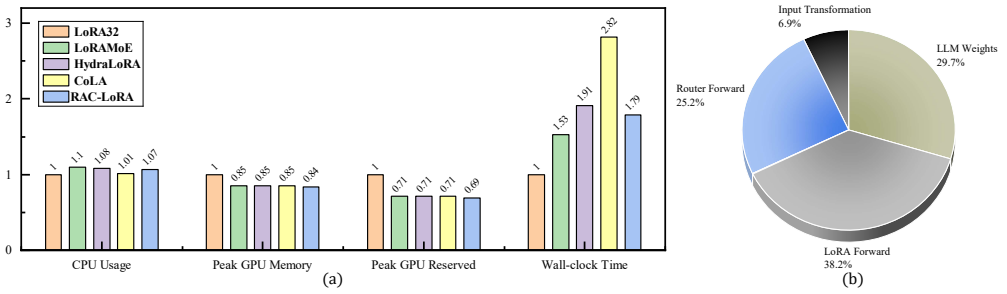


Figure 5: (a) Comparison of efficiency with standard and other variants. (b) Break down analysis of RAC-LoRA's individual components.

RAC-LoRA introduces additional computation compared with standard LoRA because it evaluates multiple implicit A/B combinations and performs router-conditioned computation. Nevertheless, its computational profile remains competitive to that of other MoE-LoRA variants at the same adapta-

tion budget. The additional cost is primarily associated with the combinatorial expert construction and router operations rather than the backbone model itself.

We further decompose the RAC-LoRA forward-pass cost into its main computational components, including routing, LoRA forwarding, router-conditioned computation, and frozen weight computation. The relative contribution of each component is shown in Fig. 5(b). This breakdown provides a more detailed view of where the additional runtime of RAC-LoRA originates and distinguishes the cost of the proposed routing mechanism from the cost of low-rank expert computation.

The efficiency analysis highlights the practical trade-off of RAC-LoRA. The proposed combinatorial and router-conditioned architecture introduces moderate computational overhead in exchange for a substantially more flexible expert space and tighter routing–expert interaction.

B Algorithmic Optimization and Pseudocode

The mathematical formulation of RAC-LoRA admits substantial computational reuse. Our implementation reorganizes the evaluation order and exploits parallel tensor operations, while preserving exactly the same forward function. No approximation is introduced.

For the standard branch, the expert computation

$$B_j A_i x \tag{25}$$

is evaluated by first computing all $A_i x$ in parallel and then reusing each result across the corresponding B_j. For the router-conditioned branch, we use

$$B_j A_i (u_{ij} u_{ij}^\top x) = (u_{ij}^\top x) B_j A_i u_{ij}, \tag{26}$$

which allows the parameter-dependent term $B_j A_i u_{ij}$ to be computed once per forward pass. Moreover, since

$$u_{ij}^\top x = \frac{w_{ij}^\top x}{\|w_{ij}\|_2}, \tag{27}$$

the required projection scalar is obtained directly from the already computed routing logits. Thus, the same logits serve both routing and conditioning without an additional projection operation.

These transformations only reorder associative matrix multiplications and reuse identical intermediate results. Therefore, the optimized implementation is algebraically equivalent to explicit expert-wise computation, while reducing redundant operations and exposing greater parallelism.

Algorithm 1 Optimized RAC-LoRA Forward Pass

Require: x, $\{A_i\}_{i=1}^{N_A}$, $\{B_j\}_{j=1}^{N_B}$, router W, λ, K

$x_{\text{in}} \leftarrow \text{Dropout}(x)$
$S \leftarrow x_{\text{in}} W^\top$
$\mathcal{M} \leftarrow \text{TopKMask}(S, K)$
$\Gamma \leftarrow \text{MaskedSoftmax}(S, \mathcal{M})$
$H \leftarrow \mathcal{M} + \Gamma - \Gamma.detach()$
$U_{ij} \leftarrow W_{ij} / \|W_{ij}\|^2$
$q_{ij} \leftarrow S_{ij} / \|W_{ij}\|^2$
$C_{ij} \leftarrow B_j A_i U_{ij}$
$G_i \leftarrow A_i x_{\text{in}}$
$F_{ij} \leftarrow B_j G_i$
$T_1 \leftarrow \sum_{i,j} H_{ij} F_{ij}$
$T_2 \leftarrow \sum_{i,j} H_{ij} q_{ij} C_{ij}$
$\Delta y \leftarrow \frac{\alpha}{r} [(1-\lambda) T_1 + \lambda T_2]$
return $y_{\text{base}} + \Delta y$

C Experimental Details

C.1 Implementation and Training Configuration

All experiments are implemented in the same PEFT-based training framework to ensure consistent backbone initialization, target modules, optimization, and evaluation procedures. We evaluate two backbone models, Llama-3.2-3B and Llama-3.1-8B. The main comparison experiments are conducted on both models, whereas all ablation and hyperparameter studies use Llama-3.1-8B.

Unless otherwise specified, the backbone parameters are frozen and only the LoRA-related parameters are optimized. We use AdamW as the optimizer with a learning rate of 5×10^{-5}. The effective batch size and the number of optimization steps are kept identical across compared methods within each experiment. The LoRA dropout is set to 0.0. All experiments are conducted on NVIDIA A800 GPUs.

Each experiment is independently repeated with three random seeds. We report the mean performance and standard deviation across the three runs. For a metric m, the reported result is

$$\bar{m} = \frac{1}{3}\sum_{k=1}^{3} m_k, \qquad \sigma_m = \sqrt{\frac{1}{3}\sum_{k=1}^{3}(m_k - \bar{m})^2}. \tag{28}$$

The shared training configuration and the method specific configuration are summarized in Table 7 and Table 8.

Table 7: Shared training configuration used across experiments.

Configuration	Value
Backbone models	Llama-3.2-3B, Llama-3.1-8B
Checkpoints	meta-llama/Llama-3.2-3B, meta-llama/Llama-3.1-8B
Hardware	NVIDIA A800
Optimizer	AdamW
Learning rate	5×10^{-5}
LoRA dropout	0.0
Evaluation metric	Accuracy
Epoch	5
Cut off length	1024
Target modules	$[q_{proj}, k_{proj}, v_{proj}, o_{proj}, up_{proj}, down_{proj}]$
Batch size	8

Table 8: Method-specific configuration of the evaluated LoRA variants. We adopt fully collaborative CoLA (CoLA$^\intercal$) as one of the comparison methods because CoLA$^\intercal$ achieves the best performance in the original paper.

Method	r	α	N_A	N_B	K	λ
LoRA	32	64	–	–	–	–
LoRAMoE	8	32	4	4	–	–
HydraLoRA	8	32	1	6	–	–
CoLA$^\intercal$	8	16	4	4	–	–
RAC-LoRA	8	32	3	3	6	0.5

C.2 Evaluation Protocol

For the general-domain setting, Databricks-Dolly-15k is used for fine-tuning and MMLU is used for evaluation. For MMLU, we report both the overall score and subject-grouped scores. The legal setting uses US-Terms and Lawyer-Instruct for training and MMLU-Law for evaluation, while the medical setting uses iCliniq and GenMedGPT-5k for training and MMLU-Medicine for evaluation.

Table 9: Trainable adapter parameters as a percentage of total backbone parameters.

Model	LoRA	LoRAMoE	HydraLoRA	CoLA	RAC-LoRA
Llama-3.2-3B	1.49%	1.58%	1.46%	1.49%	1.32%
Llama-3.1-8B	1.03%	1.09%	1.03%	1.03%	0.91%

GSM8K and FineVal-EN are evaluated on their corresponding standard test sets. Finally, OpenOrca is used for multi-task adaptation and BBH is used as the evaluation benchmark.

C.3 Ablation Configuration

All ablation experiments use Llama-3.1-8B with Databricks-Dolly-15k as the training set and MMLU as the evaluation benchmark. Unless the corresponding component is being studied, all RAC-LoRA configurations remain fixed at $N_A = N_B = 3$, $K = 6$, $\lambda = 0.5$, $r = 8$, and $\alpha = 32$.

C.4 Fairness Considerations

To ensure a fair comparison, we control the trainable parameter budget across different adaptation methods as closely as possible. In particular, the number of LoRA factors or experts is selected such that different methods have comparable numbers of trainable adapter parameters. For RAC-LoRA, we adjust the numbers of A and B factors, i.e., N_A and N_B, to achieve a comparable parameter budget with the baseline methods. Therefore, the performance differences are less likely to be attributed simply to a larger number of trainable parameters.

Table 9 reports the proportion of trainable adapter parameters relative to the total parameters of the corresponding backbone models.

D Dataset Details

We evaluate RAC-LoRA on a diverse collection of instruction-tuning and reasoning benchmarks spanning general instruction following, domain-specific knowledge, mathematical reasoning, and multi-task reasoning. Unless otherwise stated, we use the standard train and evaluation splits provided by each dataset. A brief description of each dataset is given below.

- **Databricks-Dolly-15k.** Dolly-15k is an instruction-following dataset containing approximately 15K human-generated instruction–response pairs covering a broad range of tasks, including question answering, summarization, classification, and creative generation. We use it as the general-domain training set and evaluate the resulting adapters on MMLU.
- **US-Terms.** US-Terms is a legal-domain corpus designed to provide terminology and knowledge related to the United States legal system. We combine it with Lawyer-Instruct to construct a domain-specific legal training set and evaluate the resulting model on the legal subjects of MMLU.
- **Lawyer-Instruct.** Lawyer-Instruct contains instruction-style examples targeting legal reasoning and legal question answering. Together with US-Terms, it provides both domain terminology and instruction-following supervision for the legal adaptation setting.
- **iCliniq.** iCliniq is a medical question-answering dataset collected from clinical consultations and covers a broad range of medical topics. We use it together with GenMedGPT-5k to construct the medical-domain training set.
- **GenMedGPT-5k.** GenMedGPT-5k is a medical instruction dataset containing approximately 5K instruction–response examples. It complements iCliniq by providing instruction-oriented medical supervision.
- **GSM8K.** GSM8K is a grade-school mathematical reasoning benchmark consisting of multi-step arithmetic word problems. We use its training split for adaptation and its standard test split for evaluation.

- **FineVal-EN.** FineVal-EN is an English evaluation-oriented dataset designed to assess general language understanding and generation capabilities. We use its training portion for adaptation and evaluate on the corresponding FineVal test set.
- **OpenOrca.** OpenOrca is a large-scale instruction-following dataset containing diverse synthetic instruction–response examples generated from a wide range of tasks. Its heterogeneous task distribution makes it suitable for evaluating whether a PEFT method can learn multiple adaptation patterns within a single model.
- **MMLU.** The Massive Multitask Language Understanding (MMLU) benchmark covers a wide range of academic and professional subjects. We report overall performance as well as results on the law and medicine subjects, denoted as MMLU-Law and MMLU-Medicine, respectively.
- **GSM8K-Test.** We use the standard GSM8K test split to evaluate mathematical reasoning after adaptation on the GSM8K training set.
- **FineVal-Test.** We use the official FineVal test split corresponding to FineVal-EN for evaluation.
- **BBH.** BIG-Bench Hard (BBH) is a collection of challenging reasoning tasks selected from BIG-Bench. It contains heterogeneous tasks involving, among others, logical reasoning, symbolic manipulation, and multi-step problem solving. We use BBH as an out-of-distribution evaluation benchmark after training on OpenOrca.