MoE^2-LoRA: When MoE Models meet MoE-style Low-Rank Adaptation

Anonymous ACL submission

Abstract

Mixture-of-Experts (MoE) architectures have been widely adopted in large language models, yet parameter-efficient fine-tuning (PEFT) for MoE models remains underexplored. Existing PEFT methods for MoE either ignore router priors with uniform adapters, reducing efficiency and risking forgetting, or rely on static expert selection, limiting per-token capacity and cross-expert feature learning. In this paper, we make the first attempt to fine-tune MoE models with MoE-style low-rank adaptation: our method, entitled MoE^2-LoRA, deeply couples the pretrained expert specialization with task-specific adaptivity via a dual-channel Routing-Conditioned Projection (RCP) module, which reuses base router activations to inform LoRA routing. We further introduce a single global LoRA expert pool shared across all layers, enabling model-wide adaptation with emergent layer-wise affinities and balanced expert utilization. MoE^2-LoRA simultaneously benefits from the advantages of prior reuse, dynamic adapter routing, and model-wide knowledge sharing. Evaluated on multiple MoE backbones with varying scales and expert granularities, MoE^2-LoRA consistently achieves state-of-the-art downstream accuracy while retaining stronger general capabilities.

1 Introduction

Mixture-of-Experts (MoE) (Jacobs et al., 1991) architectures, which activate only a small fraction of parameters for each input, have recently been widely adopted by large language models (LLMs) such as DeepSeek-V3 (Liu et al., 2024a) and Qwen3 (Yang et al., 2025), owing to their advantages including reasoning efficiency and scaling performance comparable to that of dense models. For dense architecture based LLMs, efficient and effective fine-tuning methods have been thoroughly studied. For example, LoRA (Hu et al., 2022), one of the most widely used parameter-efficient fine-tuning (PEFT) methods designed for dense models, matches the performance of full fine-tuning by decomposing weight updates into low-rank subspaces. In comparison, how to efficiently fine-tune MoE models remains an underexplored problem, primarily due to their sparse activation nature, unstable gradient flow, and massive parameter scale, which highlights the pressing need for PEFT methods specifically tailored for MoE models.

Existing PEFT methods for MoE models are predominantly built upon applying LoRA adapters to MoE experts (Fig. 1(I.a)), and further attempt to exploit the routing priors of the MoE model to guide expert-level adaptation, that is, to decide which experts should receive adaptation or how much (Fig. 1(I.b)). For example, ESFT (Wang et al., 2024) and DAS-LoRA (Tang et al., 2026) first collect offline router statistics to identify a subset of critical experts, and then apply LoRA only to those experts; DR-LoRA (Deng et al., 2026) adaptively assigns higher ranks to experts with higher routing frequency and gradient-based importance. While these methods improve parameter efficiency, they still suffer from two major limitations: (i) Insufficient adaptation of critical experts. By selecting a subset of trainable experts or high-rank experts, any expert who falls outside this subset but is activated at inference time receives insufficient task-specific updates. (ii) Lack of high-level interactions: Adapting experts independently neglects both the cooperative dynamics among co-activated experts and the cross-layer interactions of MoE models (Fig. 5), which could enhance performance and reduce parameter redundancy.

To address these limitations, we seek to fine-tune MoE models with dedicated MoE mechanisms, shifting the adaptation horizon from individual experts to each MoE module and the entire MoE model. This direction resonates with another line of research that integrates LoRA with an MoE structure (Fig. 1(I.c)), which yields MoE-style LoRA methods such as MoELoRA (Luo et al., 2024) and

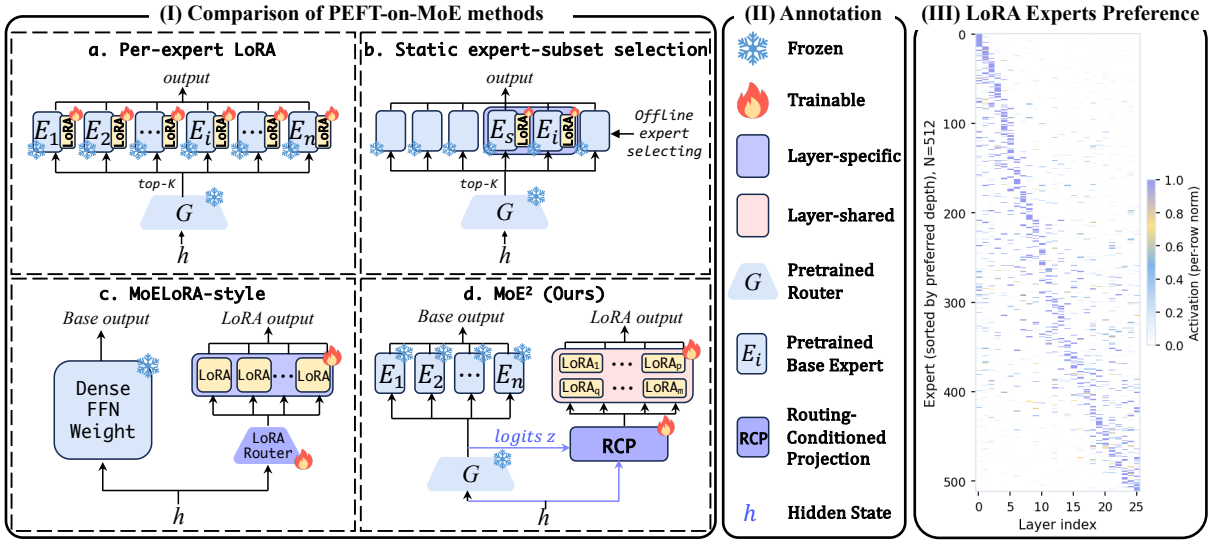

Figure 1: (I) Comparison of PEFT-on-MoE methods: (a) Per-expert LoRA, which equip each MoE expert with a LoRA adapter; (b) Static expert-subset selection methods, which select a subset of experts offline to be trained with LoRA; (c) MoELoRA-style methods, which adapt each MoE module with a MoE-style low-rank adapter; (d) MoE^2. (II) Annotation. (III) LoRA Experts Preference: We present the preferred layers of LoRA experts in MoE^2's global expert pool, demonstrating the layer-specific learning and cross-layer learning capabilities of MoE^2.

MoLA (Gao et al., 2025). These methods, however, are not specifically designed for MoE models: their MoE mechanisms operate independently of those inherent to MoE models, thereby leaving the expert interactions and cross-layer interactions encoded in the base MoE models unutilized. In light of this, if it is possible to deeply bind the two MoE mechanisms of MoE models and MoE-style LoRA, we could mitigate the aforementioned limitations. These observations naturally raise a unifying question: ***When MoE models meet MoE-style low-rank adaptation, how can we deeply bind the two MoE mechanisms at both the MoE module level and the MoE model level, thereby enhancing task performance while preserving general capabilities?***

To answer this question, we introduce MoE^2-LoRA (Figure 1(I.d)), an MoE-style low-rank adaptation method tailored for MoE models. At its core lies the Routing-Conditioned Projection (RCP) module, which reuses the pretrained MoE routing activations to drive LoRA expert selection. Concretely, RCP fuses two routing signals: a primary channel z carrying the base router activations, along with an auxiliary channel h_c, which injects a task-specific correction. A learnable matrix W_l maps the concatenated input into the LoRA routing space, producing LoRA expert selection coupled with the base model's expert selection. Further, in order to enable model-level knowledge sharing, we maintain a **global pool** of LoRA experts shared across all MoE layers. As demonstrated in Figure 1(III), this design yields a striking emergent property: each LoRA expert develops a pronounced layer affinity, while still contributing to other layers to a lesser extent, and the overall assignment of experts across layers remains highly uniform. This indicates that the global pool achieves model-wide cross-layer knowledge sharing with high parameter utilization, avoiding the isolation and redundancy of layer-specific adapters.

As a result, our MoE^2-LoRA unifies three key properties: pretrained routing prior reuse, MoE-style LoRA expert selection, and model-level cross-layer learning through a global expert pool. Equipped with this model-level design, MoE^2-LoRA attains state-of-the-art downstream accuracy on four MoE backbones of varying scale and expert granularity, OLMoE-1B-7B (Muennighoff et al., 2024), DeepSeek-V2-Lite (Liu et al., 2024a), Qwen3-30B-A3B (Yang et al., 2025), and Qwen3.5-35B-A3B (Qwen Team, 2026), improving over the strongest PEFT baselines by up to +2.56 points in in-domain average accuracy, while exhibiting superior general capability retention. Our contributions are summarized as follows:

- We diagnose two key limitations of existing PEFT methods for MoE models and introduce the principle of fine-tuning MoE models with MoE-style PEFT, which requires deeply binding the two MoE mechanisms and establish-

ing both module level and model level interactions. This principle unifies expert-level, module-level, and model-level adaptation, overcoming the identified limitations and serving as the foundation for our method design.

- We propose MoE^2-LoRA, which realizes this insight through two key components: (1) a Routing-Conditioned Projection (RCP) module that repurposes the base router's activations to drive LoRA expert selection, and (2) a single global LoRA expert pool shared across all MoE layers. As visualized in Figure 1(I), the global pool yields emergent layer affinities and balanced expert utilization, achieving efficient model-wide knowledge sharing.
- Across four MoE backbones (OLMoE-1B-7B, DeepSeek-V2-Lite, Qwen3-30B-A3B, Qwen3.5-35B-A3B) and various benchmarks (math, code, general, multimodal), MoE^2-LoRA achieves state-of-the-art downstream performance and general capability retention, improving over the base model in both task and general performance.

2 Related Work

2.1 PEFT and LoRA.

The scaling of LLMs to billions of parameters has rendered conventional full fine-tuning computationally prohibitive. This bottleneck has spurred the widespread adoption of PEFT techniques, epitomized by LoRA, owing to its plug-and-play deployability, strong downstream performance, and zero inference latency. By restricting updates to a small subset of trainable parameters, these methods enable resource-efficient model adaptation while preserving the model's original inference cost. Though LoRA matches the performance of full fine-tuning in many scenarios, a gap persists on challenging tasks such as scientific reasoning and coding. This gap has been extensively investigated and effectively narrowed by a recent body of follow-up work that refines LoRA with improved algorithms. For example, LoRA+ (Hayou et al., 2024) decouples the learning rates for the down- and up-projection weights of a low-rank adapter, effectively enhancing training stability over standard LoRA. GoRA (He et al., 2025) adaptively assigns ranks to adapters based on gradient information, thereby strengthening parameter utilization. Similarly, methods such as MoELoRA (Luo et al., 2024), MoLA (Gao et al., 2025), and LoRAMoE (Dou et al., 2023) improve LoRA's performance by integrating MoE mechanisms.

2.2 PEFT for MoE models.

PEFT methods are predominantly implemented and evaluated on dense models, without being tailored to MoE architectures. Existing studies on PEFT methods for MoE models remain sparse and insufficiently comparative. Though PERFT (Liu et al., 2026) has briefly investigated various strategies for training MoE models with LoRA, most PEFT methods for MoE models concentrate on selecting experts to be fine-tuned or adaptively assigning ranks to the adapters of MoE experts based on routing statistics and expert importance information. For example, ESFT (Wang et al., 2024) analyzes the router's output probability distribution on downstream task data and accordingly selects a subset of experts to be fine-tuned; MoE-Sieve (Manzoni, 2026) counts how many tokens each expert receives per layer on a small calibration dataset, then applies LoRA adapters exclusively to the selected experts; CEFT (Bai et al., 2025) identifies experts that progressively amplify attention to relevant contextual information and selectively fine-tunes only these experts; DR-LoRA (Deng et al., 2026) assigns heterogeneous LoRA ranks across experts based on an expert saliency score that jointly considers routing frequency and gradient-based rank importance.

3 Method

3.1 MoE Routing and LoRA

An MoE layer contains a router $G(\cdot)$ and N_E feed-forward experts $\{E_i\}_{i=1}^{N_E}$. Given a token representation $h \in \mathbb{R}^D$, the router produces logits $z = G(h)$ and activates a top-K expert set $\mathcal{S}_h$. The MoE output is

$$\mathrm{MoE}(h) = \sum_{i \in \mathcal{S}_h} g_i(h) E_i(h),$$

where $g_i(h)$ is the normalized routing weight.

LoRA parameterizes a low-rank update to a pretrained weight $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ as

$$\Delta W = \frac{\alpha}{r} BA,$$

where $A \in \mathbb{R}^{r \times d_{\text{in}}}$, $B \in \mathbb{R}^{d_{\text{out}} \times r}$, and $r \ll \min(d_{\text{out}}, d_{\text{in}})$. In PEFT-on-MoE, the pretrained router and experts are frozen, and only the introduced adapter parameters are optimized.

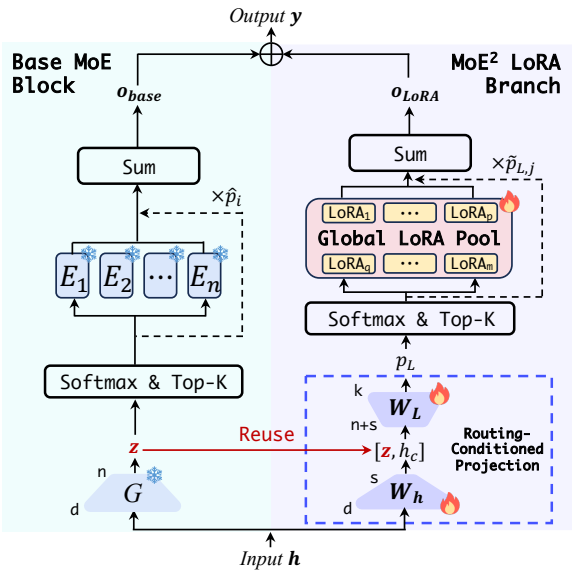

Figure 2: Overview of the MoE^2-LoRA forward pass on a single MoE layer.

3.2 Routing-Conditioned Projection (W_l)

MoE^2-LoRA introduces an additional LoRA expert pool $\{E_j^{\text{LoRA}}\}_{j=1}^{N_L}$ and learns per-token routing $p_L \in \mathbb{R}^{N_L}$ over this pool. The key design choice is the *routing input*: rather than learning routing from raw hidden state h or hardwiring it to base routing, MoE^2-LoRA uses a *two-channel input*: (1) Primary channel: the base router logits $z = G(h) \in \mathbb{R}^{N_E}$. This provides a directly pre-trained routing signal for LoRA expert selection instead of learning routing entirely from scratch; (2) Auxiliary channel: a low-rank projection of the hidden state, $h_c = W_h \cdot h \in \mathbb{R}^{d_b}$ with $d_b \ll D$. This adds a learn from scratch signal that complements the base router channel. The LoRA routing is computed as:

$$p_L = \text{softmax}\big(W_l \cdot [\,z\,;\,h_c\,]\big), \qquad (1)$$

where $W_l \in \mathbb{R}^{N_L \times (N_E + d_b)}$ is a learnable per-layer projection matrix. We then select the top-K_L LoRA experts, $\mathcal{S}_h^{\text{LoRA}} = \text{TopK}(p_L)$, and renormalize their weights to sum to one: $\tilde{p}_{L,j} = p_{L,j} / \sum_{j' \in \mathcal{S}_h^{\text{LoRA}}} p_{L,j'}$. Fig. 2 illustrates the complete forward pass.

This design conditions LoRA routing on pre-trained routing signals while preserving the flexibility of task-specific routing adaptation through the learnable projection W_l.

3.3 Global Shared LoRA Pool

MoE^2-LoRA's LoRA expert pool is *shared across layers*: a set of N_L LoRA experts is reused by multiple MoE blocks, with each block accessing the pool through its own per-layer W_l.

This contrasts with heuristic allocation-based designs such as MoLA, which modifies layer heterogeneity by *statically pre-specifying* each layer's expert pool size and composition, an architectural choice that requires search over per-layer capacity schemes. MoE^2-LoRA instead handles layer heterogeneity *dynamically at the routing level*: each layer can flexibly select from the shared pool through its learnable W_l, with no per-layer capacity hyperparameter. The same expert can serve multiple layers, and the same layer can route differently from its neighbors.

An evident advantage of cross-layer expert sharing is that the parameter count of the trainable LoRA experts pool do not scale linearly with model depth, in contrast to per-layer pool designs where the parameter count grows as $L \times N_{\text{per-layer}}$. The choice of the total number of LoRA experts and the number of activated experts per input and per layer is clearly more flexible than that of the per-layer pool designs.

3.4 Forward Pass

Combining the RCP routing of §3.2 with the global shared pool of §3.3, the MoE^2-LoRA block output is

$$y = \text{MoE}(h) + \sum_{j \in \mathcal{S}_h^{\text{LoRA}}} \tilde{p}_{L,j} \cdot E_j^{\text{LoRA}}(h), \qquad (2)$$

where $E_j^{\text{LoRA}}(h) = (\alpha/r)\, B_j A_j h$ is the rank-r LoRA expert, and $\tilde{p}_{L,j}$ are the renormalized top-K_L routing weights defined in §3.2 (or its hierarchical variant when $G > 1$).

4 Experiments

In this section, we evaluate MoE^2-LoRA and baseline methods on multiple MoE backbones across various tasks (math, code, general-retention, and multimodal).

4.1 Main Experiments

Base Models. We conduct the main experiments on three MoE backbones: **OLMoE-1B-7B** (Muennighoff et al., 2024), **DeepSeek-V2-Lite** (DeepSeek-AI, 2024), and **Qwen3-30B-A3B** (Yang et al., 2025), covering different model scales and MoE architectures. This wide set of models ensures a comprehensive evaluation across scales and model families.

Baselines. We compare MoE^2-LoRA against five baselines covering the dominant families of PEFT-for-MoE: PERFT-E (Liu et al., 2026) (per-expert LoRA following the base routing), MoELoRA (Luo et al., 2024) (per-layer LoRA expert pool with an independent learned router), MoLA (Gao et al., 2025) (heuristic expert allocation for MoE-style LoRA), DAS-LoRA (Tang et al., 2026) (DAS-guided static selection of a subset of base experts for adaptation), and FFT (full fine-tuning). All methods are applied only to the MoE modules while keeping all other modules frozen.

Datasets. We organize datasets by domain. *Math:* fine-tune on MetaMathQA-R1 (Yu et al., 2024); evaluate on GSM8K (Cobbe et al., 2021) and MATH-500 (Lightman et al., 2023). *Code:* fine-tune on MagicCoder-OSS (Wei et al., 2023) (75K samples); evaluate on MBPP (Austin et al., 2021) and HumanEval (Chen et al., 2021). *General retention:* evaluation on five general-domain benchmarks (MMLU(Hendrycks et al., 2021), WinoGrande(Sakaguchi et al., 2021), ARC-Challenge(Clark et al., 2018), StrategyQA(Geva et al., 2021), and CommonsenseQA(Talmor et al., 2019)) using the code fine-tuned checkpoints to measure general capability retention after domain-specific adaptation.

Fair comparison. We keep all essential evaluation criteria identical for all evaluated baselines: sharing the same training and testing recipe. For all evaluated methods except FFT, the trainable parameter budgets vary by at most $\sim 10\%$. (per-method counts and configurations are listed in Appendix A.2, and training-time and memory comparisons with PEFT baselines are reported in Appendix A.4).

4.2 Main Experimental Results

Tab. 1 reports in-domain performance (Math and Code) together with out-of-domain general retention across three MoE backbones.

Across all three backbones, MoE^2-LoRA achieves the highest in-domain average among PEFT methods while maintaining strong general capability retention after domain-specific fine-tuning. The gains remain consistent across models with different scales and expert granularities, ranging from OLMoE-1B-7B to Qwen3-30B-A3B. In particular, MoE^2-LoRA improves over prior PEFT-on-MoE methods on both math and code benchmarks simultaneously, whereas several baselines show a stronger trade-off between in-domain adaptation and out-of-domain retention.

4.3 Multimodal Experiments

To evaluate whether MoE^2-LoRA remains effective in multimodal settings, we further conduct experiments on a vision-language MoE model in the medical imaging domain.

Setup. We evaluate MoE^2-LoRA and two representative baselines on **Qwen3.5-35B-A3B** (Qwen Team, 2026). For in-domain medical VQA (Vision Question Answering) we fine-tune on a mixture of LLaVA-Med (Li et al., 2023) and the train splits of three medical VQA datasets, and evaluate on the test splits of VQA-RAD (Lau et al., 2018), SLAKE (Liu et al., 2021), and PathVQA (He et al., 2020). For *general VL retention*, we evaluated the checkpoints trained on medical VQA datasets on MMBench (Liu et al., 2024b), MME (Fu et al., 2025), and RealWorldQA (xAI, 2024).

Results. Tab. 2 reports both axes. MoE^2-LoRA achieves the best accuracy on every in-domain benchmark and the best positive transfer on every general benchmark, demonstrating the effectiveness of MoE^2-LoRA on multimodal settings.

4.4 Experiments on Capacity Scaling

We study scaling behavior under different trainable parameter budgets (0.12%, 0.24%, and 0.48%) on OLMoE math (Tab. 3). As the trainable budget increases, MoE^2-LoRA consistently improves on GSM8K, rising from 0.3707 at 0.12% to 0.3980 at 0.48%. In contrast, MoELoRA shows limited gains under larger budgets, particularly on GSM8K, where performance remains around 0.30 across all settings.

The results show that MoE^2-LoRA maintains stronger scaling behavior than MoELoRA under increased adaptation capacity.

5 Analysis and Discussion

In this section, we first implement ablation studies on MoE^2-LoRA to evaluate the effectiveness of its components. We then conduct in-depth analyses on the working mechanisms of the RCP and the global expert pool, uncovering the underlying advantages of our designs.

5.1 Component Ablation

We isolate MoE^2-LoRA's two architectural choices, routing-conditioned projection (RCP) and the

Method	In-domain (Math + Code)					General Retention					
	GSM8K	MATH	MBPP	HE	**Avg**	MMLU	WG	ARC-C	SQA	CSQA	**Avg**
OLMoE-1B-7B(Muennighoff et al., 2024)											
Base	1.06	0.2	23.01	14.23	9.63	45.08	51.22	52.84	56.74	45.54	50.28
FFT	46.70	17.2	28.79	21.75	28.61	45.01	50.67	52.05	53.42	48.48	49.93
per-expert LoRA(Liu et al., 2026)	29.11	7.6	28.79	14.43	19.98	48.44	50.99	56.83	52.11	47.58	51.19
MoLA(Gao et al., 2025)	26.38	7.8	29.18	15.24	19.65	**48.58**	51.22	57.42	50.80	46.68	50.94
MoELoRA(Luo et al., 2024)	29.91	5.6	29.57	15.85	20.23	47.71	51.46	56.48	**53.86**	45.21	50.94
DAS-LoRA(Tang et al., 2026)	34.87	8.2	29.96	16.46	22.37	47.82	**51.54**	57.68	52.84	45.37	51.05
MoE^2-LoRA	**36.32**	**11.2**	**31.91**	**16.87**	**24.07**	48.50	51.46	**58.19**	53.28	46.60	**51.61**
DeepSeek-V2-Lite(DeepSeek-AI, 2024)											
Base	3.71	0.0	26.46	18.90	12.27	46.09	49.96	55.03	49.34	42.59	48.60
FFT	65.05	26.0	47.86	40.45	44.84	51.55	52.17	64.42	49.49	50.12	53.55
per-expert LoRA(Liu et al., 2026)	49.28	14.4	47.86	32.32	35.97	51.59	51.30	64.85	49.20	49.80	53.35
MoLA(Gao et al., 2025)	46.25	13.8	49.03	**34.15**	35.81	51.60	50.12	64.33	49.20	**52.42**	53.53
MoELoRA(Luo et al., 2024)	48.52	14.6	49.81	33.13	36.52	51.22	50.91	64.42	**51.53**	49.96	53.61
DAS-LoRA(Tang et al., 2026)	46.55	14.6	49.03	33.54	35.93	49.64	51.54	61.77	51.09	49.80	52.77
MoE^2-LoRA	**52.24**	**18.4**	**52.14**	33.54	**39.08**	**51.72**	**51.85**	**65.53**	50.22	52.01	**54.27**
Qwen3-30B-A3B(Yang et al., 2025)											
Base	73.69	30.6	69.26	63.41	59.24	77.77	71.82	92.83	72.49	85.34	80.05
FFT	95.30	74.40	72.76	82.52	81.23	74.46	71.35	92.75	61.86	81.98	76.48
per-expert LoRA(Liu et al., 2026)	94.47	71.60	72.37	82.11	80.14	75.05	72.38	93.34	70.16	84.03	78.99
MoLA(Gao et al., 2025)	95.07	72.60	71.98	82.11	80.44	73.79	**72.77**	92.24	71.03	82.72	78.51
MoELoRA(Luo et al., 2024)	94.84	72.00	73.54	**82.52**	80.73	68.17	71.03	86.01	72.05	75.43	74.54
DAS-LoRA(Tang et al., 2026)	94.69	73.20	73.93	81.71	80.88	69.18	71.11	87.80	**73.65**	84.03	77.15
MoE^2-LoRA	**95.22**	**74.40**	**74.32**	82.32	**81.57**	**76.21**	71.51	**93.60**	70.01	**85.83**	**79.43**

Table 1: Main results across three MoE backbones on in-domain tasks (Math, Code) and out-of-domain general retention. Avg denotes the combined in-domain average over GSM8K, MATH, MBPP, HumanEval. Bold indicates the best PEFT result excluding FFT.

Method	Medical VQA				General Retention			
	VQA-RAD	SLAKE	PathVQA	**Avg**	MMBench	MME	RealWorldQA	**Avg**
Qwen3.5-35B-A3B(Qwen Team, 2026)								
Base model	69.32	72.28	50.47	64.02	90.14	91.03	46.41	75.86
MoELoRA(Luo et al., 2024)	68.20	87.30	64.82	73.44	91.61	92.04	46.14	76.60
DAS-LoRA(Tang et al., 2026)	68.60	87.40	64.60	73.53	91.61	92.38	46.54	76.84
MoE^2-LoRA	**69.25**	**87.68**	**65.67**	**74.20**	**92.19**	**92.42**	**47.19**	**77.27**

Table 2: Medical VQA accuracy and general VL retention on Qwen3.5-35B-A3B.

Method	Params	GSM8K	MATH-500
MoELoRA	0.12%	29.91	5.6
MoELoRA	0.24%	30.10	8.4
MoELoRA	0.48%	30.02	9.4
MoE^2-LoRA	0.12%	37.07	8
MoE^2-LoRA	0.24%	39.58	9.2
MoE^2-LoRA	0.48%	39.8	10.6

Table 3: Capacity scaling on OLMoE math. MoE^2-LoRA scales more effectively than MoELoRA.

shared global pool,through a matched-budget ablation chain on DeepSeek-V2-Lite (math). Starting from a MoELoRA-style baseline, we first replace hidden-state routing with RCP while keeping the per-layer pool fixed, and then introduce the globally shared pool on top of RCP.

Tab. 4 shows the main ablation chain (top three rows) together with additional reference variants (bottom). Replacing hidden-state routing with RCP under a fixed per-layer pool improves Avg from 31.56 to 32.23, showing that incorporating pretrained router structure improves LoRA expert selection even without cross-layer sharing. Introducing the globally shared pool on top of RCP further improves Avg to 35.32, indicating that global sharing and routing-conditioned projection are complementary.

We further ablate the h_c (Auxiliary channel) by routing only from pretrained router logits $z \in \mathbb{R}^{N_E}$. Both router-only variants underperform the full RCP design, indicating that pretrained routing structure alone is insufficient. Effective LoRA rout-

Variant	Pool	Routing	GSM8K	MATH-500	Avg
Baseline	per-layer	hidden-state	48.52	14.6	31.56
+ RCP	per-layer	RCP	49.05	15.4	32.23
+ Global Pool (MoE^2-LoRA)	global	RCP	**52.24**	**18.4**	**35.32**
Global Only	global	hidden-state	48.07	16.2	32.14
Router-only (per-layer)	per-layer	z only	45.49	**16.8**	31.15
Router-only (global)	global	z only	46.17	15.8	30.99

Table 4: Component ablation on DeepSeek-V2-Lite math.

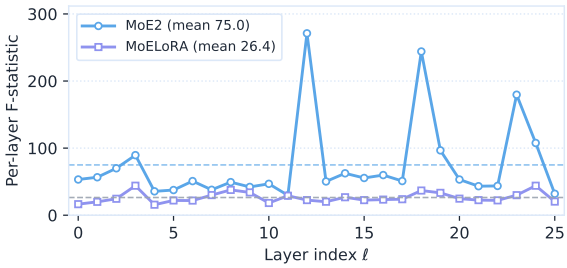

Figure 3: Per-layer routing-alignment F-statistic on DeepSeek-V2-Lite: MoE^2-LoRA (red) versus the independent-routing baseline (gray).

ing requires combining inherited router priors with input-dependent adaptation signals from the hidden state.

5.2 Analysis on RCP

To verify that RCP's per-layer projection W_ℓ actually inherits information from the base MoE router rather than learning an independent routing function, we measure the F-statistic of structural alignment between LoRA routing and base routing, with details provided in Appendix B.2. A larger F-statistic indicates that LoRA expert selection varies more systematically with the base router's expert assignments, and therefore better reflects the routing structure learned by the pretrained MoE model.

On DeepSeek-V2-Lite, MoE^2-LoRA achieves $F = 75.0$, compared with $F = 26.4$ for MoELoRA, yielding a $2.8\times$ margin. Fig. 3 further breaks down this comparison across layers. MoE^2-LoRA obtains a higher F-statistic on all 26 layers. This layer-wise consistency suggests that the alignment does not come from a few isolated layers, but is a systematic effect of conditioning LoRA routing on pretrained router logits.

These results show that RCP produces LoRA routing patterns that are substantially more aligned with the base MoE routing structure than MoELoRA's independent router. This provides direct evidence that the pretrained router logits are effectively reflected in adapter selection, supporting the intended role of RCP as a router-conditioned projection rather than an independently learned LoRA router.

5.3 Analysis on Global Pool

The shared pool architecture imposes no explicit depth constraint on expert usage. We analyze the resulting organization of the trained pool on DeepSeek-V2-Lite, focusing on its layer-aware depth organization (§5.3.1), non-uniform capacity allocation (§5.3.2), and the cross-layer representational overlap that may support such sharing (§5.3.3).

For the trained-model probes, we run forward passes of the trained MoE^2-LoRA on 300 GSM8K test prompts and collect the set T of all token positions. For each layer ℓ and LoRA expert e, we record the top-K_L selection frequency

$$f_{\ell,e} = \frac{1}{|T|} \sum_{t \in T} \mathbb{1}[e \in \mathcal{S}_{h_t}^{\mathrm{LoRA}}],$$

where $\mathcal{S}_{h_t}^{\mathrm{LoRA}}$ is the activated LoRA set for token t at layer ℓ (§3.2). We use the discrete top-K_L frequency rather than the soft routing weight to align with the experts that the model actually engages at inference.

5.3.1 Global Pool Learns Layer-aware Expert Assignment

A potential concern of a shared global LoRA pool is that experts may be used in an unstructured manner across layers, without reflecting layer-specific adaptation behavior. We therefore analyze whether the learned routing exhibits depth-dependent organization. For each expert e, we compute its preferred depth

$$\bar{\ell}_e = \frac{\sum_\ell \ell \cdot f_{\ell,e}}{\sum_\ell f_{\ell,e}},$$

where $f_{\ell,e}$ denotes the activation frequency of expert e at layer ℓ. Experts are then sorted by $\bar{\ell}_e$.

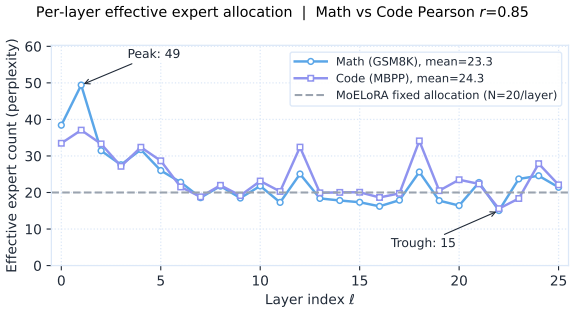

Figure 4: Per-layer effective expert count on math (blue) and code (orange) probes. Dashed line: the fixed allocation used by MoELoRA.

Fig. 1(III) visualizes the activation matrix $f_{\ell,e}$ defined above, with MoE layers on the x-axis, LoRA experts sorted by preferred depth $\bar{\ell}_e$ on the y-axis, and color intensity denoting activation frequency. The resulting diagonal structure, quantified by a Spearman correlation of $\rho = 0.92$ between $\bar{\ell}_e$ and each expert's most frequently activated layer, indicates that expert usage is depth-structured rather than layer-agnostic.

Notably, the resulting structure is localized rather than strictly layer-exclusive. Individual experts typically concentrate their activation mass within a narrow band of neighboring layers, instead of being activated by only a single layer. As a result, the global pool simultaneously preserves layer-dependent specialization and enables cross-layer expert reuse.

Appendix B.1 shows that this depth-organized structure remains consistent across different pool sizes $N_L \in \{128, 256, 512\}$ and across both math and code probes (Figs. 6, 7).

5.3.2 Per-layer Expert Capacity Is Non-uniform

For each layer ℓ, we compute the effective expert count as the perplexity of its usage distribution, $\exp(H(f_\ell))$. Fig. 4 shows that the learned allocation is highly non-uniform: the effective expert count ranges from ≈ 15 in the deepest layers to ≈ 49 in the shallowest layers, a $3.3\times$ variation.

This result suggests that adaptation demand varies substantially across depth, which is difficult to capture with fixed per-layer allocation. MoELoRA assigns the same number of experts to each layer (N=20; dashed line), while MoLA requires manually specifying a layer-wise schedule such as bottom-heavy, top-heavy, or hourglass. In contrast, MoE^2-LoRA exposes no explicit architectural knob for per-layer capacity, yet learns a non-uniform allocation automatically.

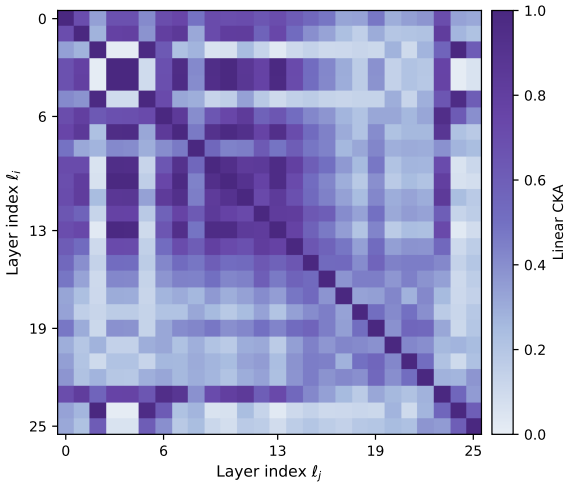

Figure 5: Linear CKA between MoE block outputs across the 26 layers of DeepSeek-V2-Lite (base model, no LoRA).

5.3.3 Cross-layer Representational Overlap Supports Pool Sharing

We further examine why sharing LoRA experts across layers is feasible. Specifically, we measure linear CKA(Appendix B.3) between MoE block outputs of every layer pair on the frozen DeepSeek-V2-Lite base model over 200 GSM8K prompts.

Fig. 5 shows clear cross-layer similarity: off-diagonal CKA averages 0.43, with adjacent layers averaging 0.56 compared with 0.35 for distant pairs ($|i-j| \geq 10$). This depth-local representational overlap provides a plausible basis for sharing LoRA experts across nearby or related layers, consistent with the soft depth-affinity pattern observed above.

6 Conclusion

We introduced MoE^2-LoRA, an MoE-style PEFT method that deeply binds the MoE mechanisms of pretrained MoE models and MoE-style low-rank adaptation through routing-conditioned projection (RCP) and a globally shared LoRA expert pool. Across four MoE backbones spanning different scales and expert granularities, MoE^2-LoRA achieves strong downstream performance while preserving general capabilities under matched training budgets. Further analysis shows that MoE^2-LoRA learns structured cross-layer adaptation while preserving alignment with pretrained MoE routing, confirming the effectiveness of its model-level design. Together, these results show that MoE^2-LoRA is an effective approach to PEFT for MoE models.

Limitations

Adapter merging is not straightforward. Like other routing-based PEFT methods such as MoELoRA, MoE^2-LoRA relies on dynamic token-wise expert selection rather than a fixed low-rank update. As a result, the learned adaptation cannot be directly merged into the base weights in the same manner as standard LoRA.

Developing mergeable approximations for dynamic routing-based PEFT architectures remains an important direction for future work.

Scaling the global pool increases projection parameter cost. The globally shared expert pool improves adaptation flexibility, but scaling the pool size also increases the parameter cost of the routing/projection interface, particularly the layer-specific projection matrices used for expert selection. A naive design that routes directly from hidden states would cause these projection parameters to grow rapidly with both hidden dimension and pool size.

MoE^2-LoRA mitigates this issue by performing routing in the pretrained router space rather than the full hidden-state space, substantially reducing projection parameter growth while preserving routing expressiveness. Nevertheless, scaling to extremely large shared pools may still require more parameter-efficient routing mechanisms.

Ethics Statement

This work introduces a parameter-efficient fine-tuning algorithm for Mixture-of-Experts language models. We discuss potential ethical considerations below.

Data. All training and evaluation data used in this paper are drawn from publicly released, well-known benchmarks (MetaMathQA, GSM8K, MATH-500, MBPP, HumanEval, MMLU, WinoGrande, ARC-Challenge, StrategyQA, CommonsenseQA). No human subjects were involved in data collection, and we performed no additional crowdsourcing or annotation. We rely on the dataset authors' license terms and use the data strictly for research purposes consistent with those terms.

Models. The base MoE models in our experiments, including OLMoE-1B-7B, DeepSeek-V2-Lite, Qwen3-30B-A3B, and Qwen3.5-35B-A3B are released under open-weight licenses for research. Our adapter weights are fine-tuned from these checkpoints and inherit any biases or limitations present in the base models. MoE^2-LoRA does not target alignment or safety properties; the routing-level adaptations we introduce do not provide any guarantee about output safety, factuality, or fairness beyond what the base model offers.

Risks of misuse. MoE^2-LoRA is a generic fine-tuning recipe and inherits the same misuse risks as any LLM PEFT method (e.g., adversarial domain adaptation, prompt-injection-resistant tuning is not addressed). It does not lower the cost of producing harmful content beyond standard PEFT, and the small adapter size is comparable to existing LoRA-family methods that are widely distributed.

Environmental impact. All experiments were conducted on a single 8-GPU node (NVIDIA H800 80GB). End-to-end fine-tuning per (method, base) pair completed within 1–2 GPU-hours on OLMoE and 1.5–3 GPU-hours on DeepSeek-V2-Lite / Qwen3-30B-A3B. The total compute reported in this paper is approximately 200 GPU-hours, dominated by FFT upper-bound runs and capacity scaling sweeps.

Reproducibility. An anonymized repository containing all training/evaluation scripts and hyperparameter configurations is included in the supplementary material. In addition, the codebase provides detailed documentation and step-by-step reproduction instructions to facilitate straightforward replication of our experiments. We will publicly release the repository and adapter weights upon acceptance. Appendix A provides a detailed reproducibility checklist including hyperparameters, training compute, data preprocessing, and evaluation protocol.

Use of AI assistants. We used AI coding assistants during code development and to assist in drafting and proofreading sections of this paper. All technical content, experimental design, results, and analyses are the authors' own; AI-assisted text was reviewed, edited, and verified by the authors.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Jun Bai, Minghao Tong, Yang Liu, Zixia Jia, and Zilong Zheng. 2025. Understanding and leveraging the expert specialization of context faithfulness in mixture-of-experts llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Peter Clark and 1 others. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

DeepSeek-AI. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*.

Guanzhi Deng, Bo Li, Ronghao Chen, Huacan Wang, Lijie Wen, and Linqi Song. 2026. Dr-lora: Dynamic rank lora for mixture-of-experts adaptation. *arXiv preprint arXiv:2601.04823*.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, and 1 others. 2023. Loramoe: Revolutionizing mixture of experts for maintaining world knowledge in language model alignment. *arXiv preprint arXiv:2312.09979*, 4(7).

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, Yunsheng Wu, Rongrong Ji, Caifeng Shan, and Ran He. 2025. Mme: A comprehensive evaluation benchmark for multimodal large language models. *Preprint*, arXiv:2306.13394.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Ruibo Liu, Baochen Sun, Yawen Zhang, Daiyi Peng, Xiaoyuan Guo, and VS Subrahmanian. 2025. Mola: Moe lora with layer-wise expert allocation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5097–5112.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies. *Transactions of the Association for Computational Linguistics (TACL)*.

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

Haonan He, Peng Ye, Yuchen Ren, Yuan Yuan, Luyang Zhou, Shucun Ju, and Lei Chen. 2025. Gora: Gradient-driven adaptive low rank adaptation. *Preprint*, arXiv:2502.12171.

Xuehai He, Yichen Zhang, Luntian Mou, Eric Xing, and Pengtao Xie. 2020. Pathvqa: 30000+ questions for medical visual question answering. *arXiv preprint arXiv:2003.10286*.

Dan Hendrycks and 1 others. 2021. 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. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Jason J Lau, Soumya Gayen, Asma Ben Abacha, and Dina Demner-Fushman. 2018. A dataset of clinically generated visual questions and answers about radiology images. *Scientific data*, 5(1):180251.

Chunyuan Li, Cliff Wong, Sheng Zhang, Naoto Usuyama, Haotian Liu, Jianwei Yang, Tristan Naumann, Hoifung Poon, and Jianfeng Gao. 2023. Llava-med: Training a large language-and-vision assistant for biomedicine in one day. *Preprint*, arXiv:2306.00890.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, and 1 others. 2024a. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*.

Bo Liu, Li-Ming Zhan, Li Xu, Lin Ma, Yan Yang, and Xiao-Ming Wu. 2021. Slake: A semantically-labeled knowledge-enhanced dataset for medical visual question answering. In *2021 IEEE 18th international symposium on biomedical imaging (ISBI)*, pages 1650–1654. IEEE.

Yilun Liu, Yunpu Ma, Yuetian Lu, Shuo Chen, Zifeng Ding, and Volker Tresp. 2026. Parameter-efficient routed fine-tuning: Mixture-of-experts demands mixture of adaptation modules. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 4439–4457, Rabat, Morocco. Association for Computational Linguistics.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua

Lin. 2024b. Mmbench: Is your multi-modal model an all-around player? *Preprint*, arXiv:2307.06281.

Tongxu Luo, Jiahe Lei, Fangyu Lei, Weihao Liu, Shizhu He, Jun Zhao, and Kang Liu. 2024. Moelora: Contrastive learning guided mixture of experts on parameter-efficient fine-tuning for large language models. *Preprint*, arXiv:2402.12851.

Andrea Manzoni. 2026. Moe-sieve: Routing-guided lora for efficient moe fine-tuning. *arXiv preprint arXiv:2603.24044*.

Niklas Muennighoff, Luca Soldaini, and 1 others. 2024. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*.

Qwen Team. 2026. Qwen3.5: Towards native multimodal agents.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158.

Yiru Tang, Kun Zhou, Xin Zhao, Jing Sha, Zhichao Sheng, and Shijin Wang. 2026. Exploring expert concentration for parameter-efficient fine-tuning of mixture-of-expert LLMs.

Zihan Wang and 1 others. 2024. Let the expert stick to his last: Expert-specialized fine-tuning for sparse architectural large language models. *arXiv preprint arXiv:2407.01906*.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. 2023. Magicoder: Empowering code generation with oss-instruct. *arXiv preprint arXiv:2312.02120*.

xAI. 2024. Grok-1.5 Vision Preview. https://x.ai/blog/grok-1.5v. RealWorldQA dataset, available at https://huggingface.co/datasets/xai-org/RealworldQA.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. In *International Conference on Learning Representations*, volume 2024, pages 45040–45061.

A Reproducibility Details

A.1 Evaluation Protocol

This appendix specifies the evaluation protocol used in all main-text experiments. The protocol is shared verbatim across MoE^2-LoRA and all baselines; only the trained adapter weights differ.

Inference configuration. The decoding configuration is shared across all methods within each benchmark; only the trained adapter weights differ. All benchmarks are evaluated in a strictly zero-shot setting without in-context examples. GSM8K, MATH-500, and MBPP use greedy decoding (`do_sample=False`, `temperature=0`), while HumanEval uses stochastic decoding with `temperature=0.2` and `top_p=0.95`. We use `max_new_tokens=1024` for all benchmarks. MBPP is evaluated with pass@1, and for HumanEval we generate n=3 samples per problem and report the average pass@1 across runs to reduce variance on the small 164-task test set.

Prompt templates. We use a single fixed prompt template per benchmark, applied identically to every method:

- **GSM8K**: `"Solve the following math problem step by step. End your answer with '#### <number>'.\n\nQuestion: {question}\nAnswer:"`
- **MATH-500**: `"Solve the following math problem step by step. Put your final answer in \boxed{...}. \n\nProblem: {problem}\nSolution:"`
- **MBPP / HumanEval**: standard function-completion prompts (problem description plus public test signature); the model is asked to produce the function body, which is then executed against the held-out unit tests.

Answer extraction. Answer extraction is identical across all methods. For GSM8K, we extract the final numeric answer using a regex match against `####\s*(-?[0-9.,]+)` and evaluate after numeric normalization. For MATH-500, we extract the contents of the first `\boxed{...}` expression (including nested braces) and evaluate after whitespace and LaTeX-symbol normalization. For MBPP and HumanEval, the generated function body is parsed from the completion and executed against the official unit tests, with pass@1 reported following the standard benchmark protocols.

A.2 Hyperparameters

MoE^2-LoRA. Table 5 summarizes the key hyperparameters used for MoE^2-LoRA across different backbones. Here, N_L denotes the size of the shared LoRA expert pool, rank the per-expert LoRA rank, top-K the number of activated LoRA experts per token, and d_b the hidden-state bottleneck dimension used in routing-conditioned projection.

MoELoRA (Luo et al., 2024). We use a per-layer LoRA expert pool with dynamic routing from hidden states. The configuration uses 16 experts (rank 8, top-2) on OLMoE-1B-7B, 32 experts (rank 6, top-4) on DeepSeek-V2-Lite, and 32 experts (rank 6, top-4) on Qwen3-30B-A3B, corresponding to approximately 8.9M, 21M, and 41M trainable parameters.

MoLA (Gao et al., 2025). MoLA uses a per-layer LoRA expert pool with a layer-wise increasing allocation, motivated by the observation that deeper transformer layers benefit from more LoRA experts. LoRA experts are routed independently from hidden states (top-2 per token). Following the original paper's allocation scheme, we use ranks 8, 22, and 40 on OLMoE-1B-7B, DeepSeek-V2-Lite, and Qwen3-30B-A3B respectively, with per-layer expert counts ranging from 8 to 24 (OLMoE, 4 stages), 4 to 16 (DeepSeek, 26 layers), and 2 to 8 (Qwen3, 4 stages). On DeepSeek-V2-Lite we scale MoLA's rank from the original 16 to 22 to match the parameter budget of other baselines. This yields approximately 8.9M, 21.4M, and 39M trainable parameters.

DAS-LoRA (Tang et al., 2026). For DAS-LoRA, we retain the original CDAS-based expert selection procedure but restrict training to the selected LoRA modules only, without additionally tuning router or dense backbone parameters, in order to maintain comparable PEFT settings across methods. Following the original paper, we pre-select a subset of base experts using Domain Advantage Score and attach LoRA modules only to the selected experts. We use rank 32 on OLMoE-1B-7B, 52 on DeepSeek-V2-Lite, and 48 on Qwen3-30B-A3B, yielding approximately 8.4M, 22M, and 38M trainable parameters.

Training. Within each backbone, all PEFT methods share the same optimizer (AdamW), cosine learning-rate schedule with warmup ratio 0.1, batch size, and number of epochs. Method-specific hyperparameters are tuned on the validation split under comparable trainable parameter budgets.

A.3 Training Compute

All experiments are conducted on a single 8×NVIDIA H800 80GB node using bfloat16 training. DeepSeek-V2-Lite and OLMoE use standard DDP, while Qwen3-30B-A3B uses DeepSpeed ZeRO-3. Typical wall-clock training time per run is approximately 35 minutes for OLMoE, 1.5 hours for DeepSeek-V2-Lite, and 2.5–3 hours for Qwen3-30B-A3B. The total compute budget for all experiments is approximately 200 GPU-hours.

Reproducibility note. Each adapter is trained once per (method, backbone) configuration. We report the per-benchmark scores from that run directly, following the evaluation protocol in Appendix A.1: a single greedy sample for GSM8K, MATH-500 and MBPP, and n=3 stochastic samples (temperature 0.2) averaged for HumanEval. Multi-run training-time statistics are not reported due to compute cost.

A.4 Training-time and memory comparability with PEFT baselines

Table 6 compares training time and peak training memory on DeepSeek-V2-Lite using 80k samples extracted from MetaMathQA for one training epoch. Training time is measured on 8× H800 80GB GPUs with DDP, while peak memory is measured on a single H800 without ZeRO using batch size 1 and sequence length 2048. All PEFT baselines are matched to roughly 22M trainable parameters; per-expert LoRA is included for reference.

All PEFT methods exhibit broadly comparable training-time and memory cost, with training time falling within a narrow 17–20 minute range and peak memory between 41.8–46.4 GB. MoE^2-LoRA lies slightly above the lightest baselines in both metrics, largely because it activates more LoRA experts per token. Compared with per-expert LoRA, the globally shared pool also avoids the larger training overhead associated with materializing independent LoRA modules for every expert.

B Additional Analysis Details

B.1 Robustness of the Depth-Organized Expert Allocation

To verify that the depth-organized diagonal observed in Section 5.3.1 is not specific to a particular configuration, we reproduce the analysis across (i)

Backbone	N_L	rank	top-K	d_b	#params
OLMoE-1B-7B	128	16	4	16	9.1M
DeepSeek-V2-Lite	256	16	6	64	21M
Qwen3-30B-A3B	256	32	6	32	39M

Table 5: MoE^2-LoRA hyperparameters across backbones. On Qwen3-30B-A3B the pool is further partitioned into G=2 groups (hierarchical routing); the other two backbones use a single flat pool (G=1).

Method	Trainable Params	Train Time (min)	Peak Train Mem (GB)
MoE^2-LoRA	21.7M	16.6	44.65
MoELoRA (Luo et al., 2024)	22.2M	19.8	43.6
MoLA (Gao et al., 2025)	21.4M	17.2	42.7
DAS-LoRA (Tang et al., 2026)	22.2M	17.8	41.8
per-expert LoRA (Liu et al., 2026)†	27.3M	23.2	44.3
FFT (DeepSeek-V2-Lite reference)‡	16B	143	120

Table 6: Training-time efficiency on DeepSeek-V2-Lite math. †Per-expert LoRA cannot be configured to a 22M budget without collapsing to rank-1; we report its native 27M configuration.

different global pool sizes and (ii) different probe domains.

Across pool sizes. Fig. 6 shows the expert-affinity matrices on GSM8K for $N_L \in \{128, 256, 512\}$. The same depth-organized diagonal appears in all three settings, indicating that experts consistently specialize to contiguous depth regions. As N_L increases, the diagonal becomes thinner because the 26-layer depth axis is partitioned among more experts; smaller pools produce correspondingly broader depth bands.

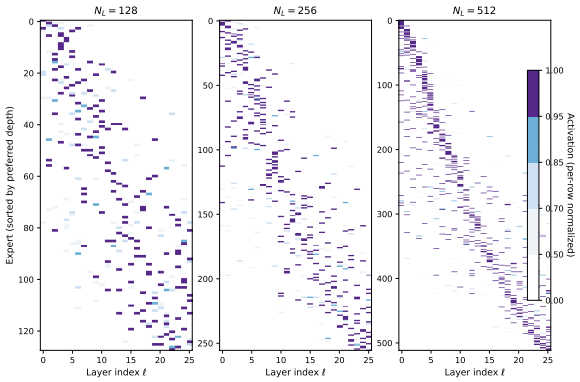

Figure 6: Expert-affinity matrices on GSM8K for MoE^2-LoRA with different global pool sizes $N_L \in \{128, 256, 512\}$ on DeepSeek-V2-Lite. Rows are experts sorted by preferred depth; columns are MoE layers. A consistent depth-organized diagonal emerges across all pool sizes.

Across domains. Fig. 7 compares GSM8K and MBPP using the same trained adapter ($N_L = 512$). Both domains exhibit the same depth-organized structure, despite the code probe appearing slightly noisier due to the smaller MBPP evaluation set.

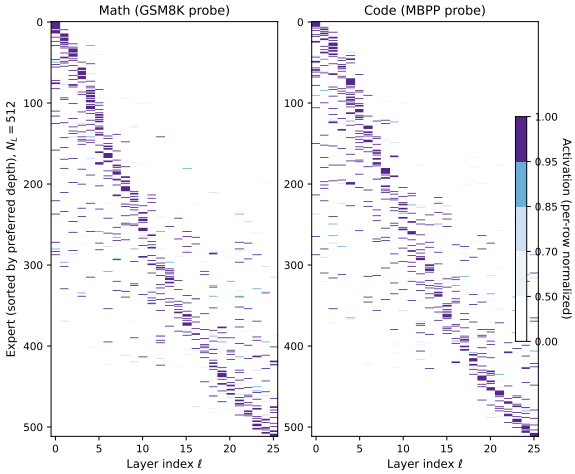

Figure 7: Expert-affinity matrices for the same trained adapter evaluated on GSM8K (left) and MBPP (right). Each panel is independently sorted by preferred depth. The depth-organized structure is preserved across domains.

Across both pool sizes and probe domains, the same depth-organized allocation consistently emerges, suggesting that it is an intrinsic property of the shared global pool with routing-conditioned projection rather than an artifact of a particular setting.

B.2 F-statistic for routing alignment

The routing-alignment F-statistic reported in Section 5.2 is a one-way ANOVA statistic that measures how much of the variance in LoRA routing

can be explained by the base MoE router's expert assignment—higher values indicate stronger structural inheritance of the pretrained routing prior.

Formally, let T denote the token set, $\mathbf{p}_L(t) \in \mathbb{R}^{N_L}$ the LoRA routing distribution for token t, and $e_b(t) = \arg\max_e \mathbf{p}_B(t)$ the base router's top-1 expert for t. We partition T into K groups indexed by the base expert, $G_e = \{t \in T : e_b(t) = e\}$, with sizes $n_e = |G_e|$ (groups with $n_e < 5$ are dropped to avoid noisy estimates). Let $\boldsymbol{\mu}_e = \frac{1}{n_e} \sum_{t \in G_e} \mathbf{p}_L(t)$ denote the group-mean LoRA distribution and $\bar{\boldsymbol{\mu}} = \frac{1}{|T|} \sum_t \mathbf{p}_L(t)$ the grand mean. The between- and within-group sums of squares are

$$\mathrm{SS}_{\text{between}} = \sum_{e=1}^{K} n_e \, \|\boldsymbol{\mu}_e - \bar{\boldsymbol{\mu}}\|_2^2, \tag{3}$$

$$\mathrm{SS}_{\text{within}} = \sum_{e=1}^{K} \sum_{t \in G_e} \|\mathbf{p}_L(t) - \boldsymbol{\mu}_e\|_2^2, \tag{4}$$

and the F-statistic is the ratio of mean squares

$$F = \frac{\mathrm{SS}_{\text{between}}/(K-1)}{\mathrm{SS}_{\text{within}}/(|T|-K)}. \tag{5}$$

A larger F indicates that LoRA routing distributions are tightly clustered within each base-expert group and well-separated across groups—i.e., the LoRA router behaves predictably as a function of the base router. We treat F as a structural-fit diagnostic for the RCP projection W_ℓ rather than a quality metric; passive coupling schemes (e.g., per-expert LoRA) attain trivially high F by construction.

B.3 Linear CKA

Given centered layer representations $X_i \in \mathbb{R}^{n \times d_i}$ and $X_j \in \mathbb{R}^{n \times d_j}$, we compute linear centered kernel alignment (CKA) as

$$\mathrm{CKA}(X_i, X_j) = \frac{\|X_i^\top X_j\|_F^2}{\|X_i^\top X_i\|_F \, \|X_j^\top X_j\|_F}.$$

Here $\|\cdot\|_F$ denotes the Frobenius norm. For each layer pair, X_i and X_j are constructed from the corresponding MoE block outputs collected over the evaluation prompts.