Beyond Symmetric LoRA: Asymmetric Parallel LoRA with Smart Routing

Anonymous ACL submission

Abstract

Mixture-of-Experts Low-Rank Adaptation (MoE-LoRA) is widely used as a paradigm to alleviate the rank gap between LoRA and full fine-tuning. However, adopting symmetric or complete asymmetric structures, existing methods often employ routing mechanisms that depend on input features solely, disregarding the heterogeneous capabilities of downstream experts. To address the problem, we introduce Asymmetric Parallel LoRA (AP-LoRA) with two key improvements: (1) Asymmetric parallel architecture enables multiple low-rank projection matrices to interact in a many-to-many fashion; (2) The Smart Router module is designed for both input features and LoRA weights, ensuring better alignment. Experiments are conducted on natural language understanding and algorithm reasoning tasks with language models ranging from 0.6B to 7B in size. The comparative results indicate that AP-LoRA outperforms existing MoE-LoRA variants, achieving state-of-the-art performance under comparable parameter and compute budgets.

1 Introduction

Low-Rank Adaptation (LoRA) (Hu et al., 2022) has emerged as a powerful paradigm in the field of parameter-efficient fine-tuning (PEFT) (Houlsby et al., 2019). By constraining task-specific updates to a low-rank subspace, LoRA significantly reduces the number of trainable parameters while maintaining strong empirical performance. However, the low-rank constraint inevitably damages the representational ability. Compared to an optimal weight matrix W obtained from full fine-tuning (FFT), LoRA is more likely to converge to a suboptimal weight matrix W'. To mitigate this issue, existing studies mainly explore two directions: modifying LoRA architecture within a limited parameter budget, and improving expressivity through parallel LoRA modules.

As a widely adopted parallel computation paradigm, Mixture-of-Experts (Shazeer et al., 2017) has been extensively incorporated into LoRA-based methods. MoE-LoRA (Zhao et al., 2024; Feng et al., 2024; Yang et al., 2024; Huang et al., 2024; Liu et al., 2024a) architectures aim to outperform standard LoRA by composing multiple low-rank experts in a structure managed by a router, rather than naively increasing the rank. Despite promising results in several studies, the practical effectiveness of MoE-LoRA remains subject to two fundamental challenges. Firstly, most existing methods focus primarily on aggregating multiple LoRA structures, while overlooking the internal structure of LoRA itself. Secondly, the router in MoE-LoRA typically maps inputs to a probability distribution over experts without explicitly modeling expert capability and specialization, limiting its effectiveness on heterogeneous task distributions.

To address the aforementioned challenges, we propose a unified framework that categorizes existing MoE-LoRA methods into symmetric and asymmetric architectures for theoretical analysis. Building upon this perspective, we introduce AP-LoRA. AP-LoRA adopts an asymmetric parallel architecture together with a smart routing mechanism, aiming to further bridge the gap between LoRA and FFT. Through a detailed analysis of the alignment problem in MoE-LoRA, we show that approximately orthogonal experts, when combined through optimal routing, can achieve better alignment than standard LoRA. Moreover, the reduction in approximation error compared with LoRA is proportional to the variance of the dynamic routing weights. The key insight lies in constructing an optimal combination of experts, together with an optimal routing strategy that realizes such combinations.

Our contributions are summarized as follows:

- **Novel Architecture.** As an alternative to con-

ventional MoE-LoRA designs, we propose an asymmetric parallel architecture, whose expert space strictly contains those of both symmetric and asymmetric architectures.

- **Smart Router.** We introduce a routing mechanism that formulates routing as an input–weight alignment problem, and enhance its expressive capability through learnable nonlinear transformations.

- **Empirical Validation.** Extensive experiments on natural language understanding and arithmatic reasoning tasks demonstrate the effectiveness and robustness of AP-LoRA with acceptable computational overhead.

2 Related Works

2.1 LoRA

LoRA assumes that the weight update during fine-tuning can be represented as a product of two low-rank matrices, which significantly reduces the number of trainable parameters. To enhance the representational capacity of LoRA, prior methods (Hu et al., 2023a; Zhang et al., 2023; Wu et al., 2024) have explored architectural modifications that lead to improved performance. DoRA (Liu et al., 2024b) further decomposes the LoRA update into a magnitude vector and a directional matrix, aligning the evolution of magnitude and direction with that of FFT. Aurora (Dong et al., 2025) introduces both fixed and learnable nonlinear transformations between the two low-rank matrices in LoRA, providing a flexible architecture.

Another stream of work investigates practical realizations of MoE-LoRA. LoRAMoE (Dou et al., 2024) adopts a simple parallel composition of multiple LoRA modules and employs a router to determine the contribution of each LoRA expert. MoLA (Gao et al., 2025) allocates different numbers of LoRA experts across layers, highlighting expert redundancy and heterogeneity across different transformer layers. AlphaLoRA (Qing et al., 2024) leverages the weight magnitudes of the backbone network to adaptively allocate the number of experts for each module, achieving memory efficiency while maintaining performance. HydraLoRA (Tian et al., 2024) proposes an asymmetric MoE-LoRA architecture, in which a single A matrix corresponds to multiple B matrices, and demonstrates its effectiveness empirically. R-LoRA (Liu et al., 2025) improves HydraLoRA by introducing randomness and refined initialization strategies, encouraging diversity among experts. FlyLoRA (Zou et al., 2025) achieves asymmetry implicitly through rank-wise routing within a single A–B pair. In addition, several studies (Meng et al., 2024; Zhang and Pilanci, 2024) investigate the alignment between LoRA and FFT from a gradient perspective. LoRA-Pro (Wang et al., 2025) enhances the update mechanism of standard LoRA, and subsequent work (Sun et al., 2025) extends this approach to the MoE-LoRA paradigm.

2.2 Mixture-of-Experts

Mixture-of-Experts (MoE) architectures have been extensively studied as an effective approach to scaling model capacity while controlling computational cost. Early works, such as sparsely-gated MoE (Shazeer et al., 2017), introduce a learned router to dynamically select a small subset of experts for each input, enabling conditional computation. Subsequent studies improve routing stability and efficiency through load-balancing objectives, expert capacity constraints, and auxiliary losses. Switch Transformer (Fedus et al., 2022) simplifies the routing mechanism by activating a single expert per token, significantly reducing communication overhead while preserving scalability. GShard (Lepikhin et al., 2021) further explore large-scale MoE training.

More recently, several studies have observed that the routing mechanism is often poorly optimized in practice. To address this issue, some approaches employ dense backpropagation (Panda et al., 2025), allowing the router to receive gradient signals from all experts when only a sparse subset is activated. The ERC loss (Lv et al., 2025a), introduced as an auxiliary objective, enforces a tight correspondence between each row of the routing matrix and a specific expert, thereby explicitly binding the router to the experts. Other works move beyond explicitly designed routing modules. AoE (Lv et al., 2025b) computes routing weights directly from a subset of expert parameters, enabling experts to implicitly determine which inputs they should handle. Similarly, Competes MoE (Pham et al., 2024) determines each expert's contribution based on its input activations.

3 Method

In this section, we firstly develop an alignment theory for the MoE-LoRA paradigm. Guided by

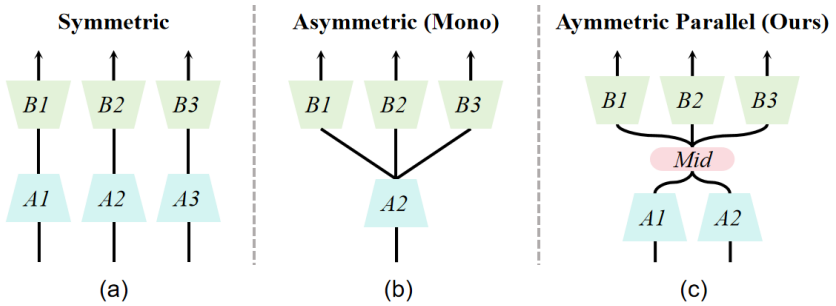

Figure 1: Comparison of different MoE-LoRA architectures. (a) Symmetric architecture. Each matrix A is manually paired with a fixed matrix B. Most existing MoE-LoRA variants fall into this category, including LoRAMoE, MixLoRA, and AlphaLoRA. Although these methods differ in expert allocation strategies, their underlying implementations follow the same symmetric design. (b) Asymmetric architecture. Recent methods such as HydraLoRA and R-LoRA adopt an asymmetric formulation, where either a single A or a single B is shared across experts. In this setting, routing is applied to only one side. (c) Asymmetric parallel architecture. The proposed architecture allows unequal numbers of matrices A and B, while routing is applied in parallel to both sides. This design enables more flexible expert combinations and a richer representation space.

the theoretical analysis, we further design two key components: the Asymmetric Parallel Architecture and the Smart Router. The Asymmetric Parallel Architecture is illustrated in Figure 1, while the Smart Router is shown in Figure 2.

3.1 Asymmetric Parallel Architecture

We first clarify a commonly misunderstood aspect of MoE-LoRA by decoupling the MoE formulation from the notion of rank.

Lemma 3.1 (Rank Limitation of MoE-LoRA)**.** *Under a fixed parameter budget with all experts simultaneously activated, MoE-LoRA cannot increase the rank of LoRA in any form.*

The formal proof is deferred to Appendix A.1. Lemma 3.1 shows that, despite using input-dependent weights, MoE-LoRA cannot break the intrinsic low-rank constraint of LoRA. Nevertheless, MoE-LoRA variants outperform standard LoRA on a wide range of tasks, indicating that the capacity of MoE-LoRA is not solely determined by rank, but also by the input-dependent composition of experts. In the following analysis, we focus on the symmetric formulation, as the asymmetric form can be viewed as a special mathematical case.

Since the representational gap between adapters manifests as missing singular directions in the weight update, we measure the approximation error via the nuclear norm $\|\cdot\|_\star$, defined as the sum of singular values $\|M\|_\star = \sum_i \sigma_i(M)$. Notably, for matrices with orthogonal column spaces, the nuclear norm is additive:

$$\|A + B\|_\star = \|A\|_\star + \|B\|_\star \quad \text{if } A \perp B.$$

Assumption 3.2 (Approximate Orthogonality)**.** Throughout our analysis, subspace refers to the column space induced by the LoRA decomposition, $S := \mathrm{span}(\mathrm{cols}(BA))$ for standard LoRA, and $S_k := \mathrm{span}(\mathrm{cols}(B_k A_k))$ for the k-th expert. In an ideal MoE-LoRA framework, these expert column spaces should be approximately orthogonal. Formally, for unit vectors $x_i \in S_i$ and $x_j \in S_j$ with $i \neq j$:

$$|\langle x_i, x_j \rangle| < \epsilon,$$

where ϵ is a negligible positive constant.

Lemma 3.3. *MoE-LoRA achieves lower approximation error by optimizing over a collection of low-rank subspaces rather than a single low-rank subspace.*

In practical fine-tuning scenarios, particularly in multi-task or compositional reasoning settings, the optimal weight update is inherently input-dependent. Distinct input patterns require adapting different functional subspaces. We therefore model the target as an input-dependent matrix $G(x) \in \mathbb{R}^{m\times n}$, which is typically high-rank. Let $W_k = B_k A_k$ denote the weight update of the k-th expert. Let $W : \mathcal{X} \to \mathbb{R}^{m\times n}$ be the mapping from inputs to dynamic weights, and $\phi_k : \mathcal{X} \to (0, 1)$ be the routing function of the k-th expert. The effective weight under MoE-LoRA is given by

$$W(x) = \sum_{k=1}^{K} \phi_k(x) W_k.$$

.

Optimality is defined as minimizing the expected

nuclear-norm approximation error:

$$\min \ \mathbb{E}_x \|G(x) - \hat{W}(x)\|_\star.$$

Standard LoRA approximates $G(x)$ with a single fixed low-rank transformation, which is insufficient when $G(x)$ varies significantly with input. MoE-LoRA instead employs an input-adaptive combination of multiple low-rank components, effectively approximating a union of subspaces. The formal derivation is given in Appendix A.3. The key result is:

$$\begin{aligned} \mathcal{L} &= Err^{\min}_{LoRA} - Err^{\min}_{MoE\text{-}LoRA} \\ &= \sum_k \mathbb{E}_x[|\phi^\star_k(x) - \alpha^\star_k|] \cdot \|W_k\|_\star, \end{aligned} \tag{1}$$

where $\alpha^\star_k$ are the optimal coefficients and $\phi^\star_k(x)$ are the optimal input-dependent routing weights. When the routing weights exhibit high variance across inputs, MoE-LoRA achieves a strictly smaller minimal approximation error than standard LoRA.

Assumption 3.2 may not hold strictly in all scenarios, but Lemma 3.3 does not require it. The result holds as long as

$$\mathbb{E}_x \left\| \sum_k (\phi^\star_k(x) - \alpha^\star_k) W_k \right\|_\star > \mathbb{E}_x \|\Delta \Pi_S G(x)\|_\star, \tag{2}$$

a condition we verify empirically across tasks in Appendix B.

Lemma 3.4. *The asymmetric parallel paradigm of MoE-LoRA has strictly stronger representational capacity than the symmetric paradigm under the same parameter budget.*

The key insight is that symmetric MoE-LoRA restricts combinations to $\{B_k A_k\}_k$, whereas the asymmetric parallel architecture spans the strictly larger family $\{B_j A_i\}_{i,j}$, without enforcing any predefined expert pairing. The formal proof is provided in Appendix A.4.

Theorem 3.5. *Combining Lemma 3.3 and Lemma 3.4, MoE-LoRA with approximately orthogonal experts achieves better alignment with the full-rank optimal matrix than standard LoRA, and the asymmetric parallel paradigm further enlarges the representational space. The conclusion holds without Assumption 3.2 provided Eq. 2 is satisfied.*

As illustrated in Figure 1, the asymmetric parallel architecture employs asymmetric sets of matrices A and B, and introduces two parallel routing mechanisms to jointly evaluate their contributions. This design no longer enforces a predefined pairing between experts and their corresponding A and B matrices; instead, task-specific compositions are formed dynamically:

$$G_{\text{mix}} = \left(\sum_j \beta_j B_j \right) \left(\sum_i \alpha_i A_i \right). \tag{3}$$

Equation 3 shows that this architecture admits greater expressive power than previous MoE-LoRA frameworks. By additionally removing the constraint that the number of A and B matrices must be identical, AP-LoRA further enlarges the representational space.

3.2 Smart Router

Lemma 3.3 establishes that, within the MoE-LoRA framework, an optimal router can significantly reduce the minimal approximation error. However, in conventional MoE architectures, the router maps input features solely to expert weights, largely ignoring the task-specific capabilities of individual experts. Prior work (Panda et al., 2025) has analyzed gradient backpropagation through the routing mechanism, which helps clarify this limitation. Specifically, the gradient of the output with respect to the router parameters can be expressed as a weighted aggregation of expert activations, indicating that the routing signal depends only on the summed responses of experts. Let π be the expert weights.

$$\frac{\partial W(x)}{\partial \phi} = \left[\sum W_1 x, \ldots, \sum W_K x \right]^T \frac{\partial \pi}{\partial \phi}.$$

We argue that relying solely on aggregated expert responses to guide routing is suboptimal. To address this issue, we introduce a Smart Router, whose architecture is illustrated in Figure 2. The key idea is to inject expert-specific features, e_i in Figure 2, into the routing process through learnable nonlinear transformations, thereby tightly coupling expert representations with routing decisions. To prevent undesirable feedback loops and ensure stable optimization, gradients from the router to expert parameters are detached.

To reduce computational overhead while maintaining representational fidelity, we employ simple yet effective expert features based on averaged encoding vectors. For the k-th A matrix, we extract the mean of its row vectors as a feature, since each

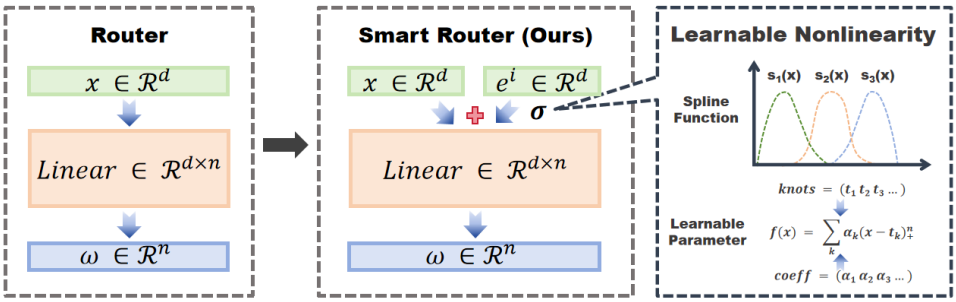

Figure 2: Visualization of Smart Router.

row serves as an encoding vector that maps inputs into real-valued space. For B matrices, we first compute the mean of the row vectors and then derive an encoding vector by weighting it with the associated A matrix, yielding an expert representation aligned with the input space.

$$a_i = \text{mean}(A_i), \quad b_i = \left(\sum_j A_j^T\right) \text{mean}(B_i).$$

To ensure the stability of AP-LoRA across diverse tasks, we adopt spline functions as the nonlinear activation in the Smart Router. All spline coefficients are initialized to zero, guaranteeing that the Smart Router performs no worse than a standard input-only router at initialization. With the router matrix $W_\Phi \in \mathbb{R}^{K\times d}$ and the expert latent matrix $E = [e^1, \ldots, e^K] \in \mathbb{R}^{d\times K}$, the weight for the i-th expert $\pi_i \in \mathbb{R}$ is calculated as:

$$\pi_i = (W_\Phi x)_i + \sum_{j=1}^{d} W_{i,j}\, \sigma(e_j^i),$$

where the nonlinear mapping $\sigma(\cdot)$ is implemented as a spline function. For AP-LoRA, expert weights are denoted α_i and β_j, with $G_{\text{mix}} = \left(\sum_j \beta_j B_j\right)\left(\sum_i \alpha_i A_i\right)$.

4 Experiments

We conduct a comprehensive set of experiments to evaluate the effectiveness of AP-LoRA. Our evaluation focuses on two representative task categories: natural language understanding (NLU) and arithmetic reasoning (AR). For NLU, we consider three datasets from the GLUE (Wang et al., 2018) benchmark and three question answering datasets. For AR, models are trained on the Math10K (Hu et al., 2023b) dataset and evaluated on six arithmetic reasoning benchmarks. Experiments are performed on three model scales ranging from 0.6B to 7B parameters, including Qwen3-0.6B, Llama3.2-3B, and Llama-7B (Touvron et al., 2023).

We further analyze the newly introduced hyperparameters in AP-LoRA, the numbers of A and B matrices, to examine the scalability of the proposed method as the parameter budget increases. To demonstrate that AP-LoRA does not introduce severe computational overheads, we perform efficiency analysis and compare AP-LoRA with representative MoE-LoRA variants. In addition, we conduct ablation studies by systematically removing individual components of AP-LoRA to assess their respective contributions.

4.1 Setup

Due to the diversity of tasks and model scales, hyperparameters vary across experiments. We describe a representative setup using Qwen3-0.6B on NLU tasks. Specifically, we adopt a batch size of 8 and a learning rate of 3×10^{-4} and fine-tune all modules for 10 epochs to achieve optimal performance.

Since MoE-LoRA variants differ substantially in their architectures, we carefully adjust the number of experts in each model to ensure comparable parameter budgets. To ensure a fair comparison across models, we fix the total number of parameters to be approximately the same as that of a LoRA model with rank 32. A LoRA model with rank 8, corresponding to one-quarter of this parameter budget, is used only to assess scalability with respect to parameter count.

4.2 Natural Language Understanding

The NLU evaluation consists of six datasets: CoLA, MRPC (Dolan and Brockett, 2005), and RTE from the GLUE (Wang et al., 2018) benchmark, as well as CommonsenseQA (Talmor et al., 2019), ScienceQA (Saikh et al., 2022), and OpenbookQA

Table 1: Performance of different rank of naive LoRA and several MoE-LoRA variants on natural language understanding tasks. A, B represent the number of matrices A and matrices B respectively. The number of two matrices of AlphaLoRA and MoLA cannot be shown because they vary number of experts across layers.

Model	Method	A	B	CoLA	MRPC	RTE	CSQA	OBQA	ScienceQA	Avg
Qwen3-0.6B	FFT	-	-	76.51	79.13	76.53	59.71	64.80	88.21	74.15
	LoRA8	1	1	75.26	79.01	76.90	59.04	64.40	87.32	73.65
	LoRA32	1	1	76.51	80.29	77.26	61.42	68.60	88.71	75.46
	FlyLoRA	1	1	76.11	79.71	76.89	62.08	69.40	88.94	75.52
	LoRAMoE	4	4	76.89	**80.64**	75.81	62.16	69.20	89.52	75.70
	AlphaLoRA	-	-	76.99	79.30	76.90	62.65	69.00	89.12	75.66
	HydraLoRA	1	6	76.41	78.32	**79.42**	62.33	68.00	89.43	75.65
	R-LoRA	1	6	76.99	79.88	77.26	63.23	68.60	89.39	75.89
	AP-LoRA(ours)	3	4	**78.62**	80.25	77.26	**65.36**	**69.40**	**89.93**	**76.80**
Llama-7b	LoRA32	1	1	83.80	83.19	82.31	75.42	78.20	89.92	82.14
	LoRAMoE	4	4	83.12	83.76	83.39	75.75	**79.20**	90.06	82.54
	MoLA-∇	-	-	82.55	83.36	83.30	75.83	77.60	90.10	82.12
	AlphaLoRA	-	-	82.42	83.19	83.13	76.17	78.20	89.88	82.17
	HydraLoRA	1	6	83.89	83.36	83.39	75.10	78.00	90.24	82.33
	R-LoRA	1	6	83.51	83.42	83.03	75.67	78.20	**90.65**	82.41
	AP-LoRA(ours)	3	4	**84.56**	**84.29**	**84.47**	**76.49**	79.00	90.60	**83.23**

(Mihaylov et al., 2018). These datasets cover a diverse range of linguistic and reasoning capabilities, including grammatical acceptability, paraphrase detection, textual entailment, and commonsense and scientific reasoning. Experiments are conducted on both the training and test splits of all datasets. For Qwen3-0.6B, models are fine-tuned for 10 epochs, as additional training leads to performance degradation due to overfitting. In contrast, Llama-7B (Touvron et al., 2023) requires 20 epochs to reach its optimal performance.

Results are reported in Table 1. As the rank increases, standard LoRA exhibits only marginal performance improvements. Specifically, LoRA with rank 32 improves over rank 8 by only 1.79%, suggesting limited scalability through rank expansion alone. Among existing MoE-LoRA variants, R-LoRA (Liu et al., 2025) achieves the strongest performance. AP-LoRA consistently outperforms all baselines, yielding an average accuracy improvement of 1.34%/1.09% over LoRA with rank 32 across NLU tasks.

4.3 Arthimatic Reasoning

For arithmetic reasoning, we follow the task formulation of previous work (Hu et al., 2023b). Models are trained on the Math10K (Hu et al., 2023b) dataset for three epochs and evaluated on six downstream benchmarks: MultiArith (Roy and Roth, 2015), GSM8K (Cobbe et al., 2021), AddSub (Hosseini et al., 2014), AQuA (Ling et al., 2017), SingleEq (Koncel-Kedziorski et al., 2016), and SVAMP (Patel et al., 2021). These benchmarks span a wide range of arithmetic reasoning skills, including multi-step computation, equation solving, and compositional numerical reasoning. We adopt the same experimental settings as the original work, with the only modification being the backbone model, which is replaced by Llama3.2-3B.

AP-LoRA achieves the highest accuracy across all arithmetic reasoning benchmarks, outperforming LoRA with rank 32 by 1.75% and exceeding the strongest existing MoE-LoRA variant, R-LoRA, by 0.59%. These results demonstrate the robustness and generality of AP-LoRA across a diverse set of tasks. While standard LoRA achieves only limited improvements through rank expansion, MoE-LoRA architectures, especially AP-LoRA, enable substantially enhance reasoning capability under comparable computational and time budgets. Overall, AP-LoRA establishes a state-of-the-art parameter-efficient adaptation framework and offers a principled direction for improving expressive capacity without significant overhead.

4.4 Hyperparameter Analysis

AP-LoRA introduces two additional hyperparameters, the numbers of matrices A and B. In this experiment, we evaluate on GSM8K to systematically study how different configurations of these two parameters affect model performance, as well as how AP-LoRA behaves when its overall parameter scale is increased or reduced.

A full grid search over both hyperparameters is computationally prohibitive. Therefore, we adopt a constrained search strategy by fixing several val-

Table 2: Arthimatic reasoning datasets. Models are trained with Llama3.2-3B on Math10k and evaluated on six downstream datasets.

Model	Method	A	B	MultiArith	GSM8K	AddSub	AQuA	SingleEq	SVAMP	Avg
Llama3.2-3B	FFT	-	-	97.83	55.72	90.63	28.35	95.47	68.60	72.77
	LoRA32	1	1	98.00	50.41	89.87	25.20	93.90	65.40	70.46
	LoRAMoE	4	4	97.67	49.13	89.62	24.80	90.55	65.30	69.51
	AlphaLoRA	-	-	98.17	50.56	91.39	26.38	93.38	67.40	71.21
	HydraLoRA	1	6	98.33	51.48	89.37	24.41	93.11	**67.80**	70.75
	R-LoRA	1	6	98.50	50.95	90.63	**28.35**	93.70	67.60	71.62
	AP-LoRA(ours)	3	4	**98.64**	**52.46**	**92.66**	27.56	**94.49**	67.50	**72.21**

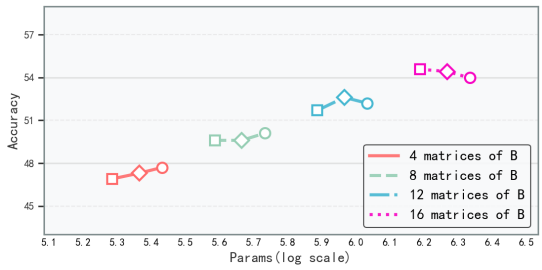

Figure 3: Visualization of Parameter Expansion. Different ratios of $|A| : |B|$ are indicated by □ (1:4), ◇ (2:4), ◯ (3:4).

ues of $|B|$ and varying $|A|$ to explore their optimal ratio. This design is motivated by observations from prior work, where matrices B are shown to distribute across diverse directions during optimization, while matrices A tend to exhibit more concentrated distributions. Based on this empirical insight, we hypothesize that the optimal configuration generally satisfies $|A| < |B|$. Accordingly, for each fixed $|B|$, we search downward to identify the best-performing $|A|$.

Figure 3 illustrates the optimal choices of $|A|$ under four different settings of $|B| \in \{4, 8, 12, 16\}$, where $|A|$ is selected from $\{\frac{1}{4}, \frac{2}{4}, \frac{3}{4}\} \cdot |B|$. When the number of B matrices is small, performance consistently peaks when $|A|$ is close to $\frac{3}{4}|B|$, suggesting that increasing the number of A matrices effectively enhances model expressiveness in low-$|B|$ regimes. In contrast, when $|B|$ becomes larger, the optimal $|A|$ shifts toward $\frac{1}{4}|B|$, and further increasing $|A|$ yields only marginal improvements.

These results indicate that as the number of B matrices increases, the expressive capacity of AP-LoRA gradually approaches saturation. In this regime, adding more A matrices provides diminishing returns, and AP-LoRA behaves increasingly similarly to an asymmetric MoE-LoRA architecture dominated by the diversity of B. Conversely, when the number of B matrices is limited, increasing $|A|$ substantially improves performance, highlighting the effectiveness of the asymmetric parallel design in compensating for limited expert diversity.

4.5 Efficiency

AP-LoRA introduces an asymmetric parallel architecture together with an smart routing mechanism. Compared to both symmetric and asymmetric MoE-LoRA variants, this design is inherently more complex, as it involves two routing functions as well as full interactions between matrices A and B. A natural concern arising from this design is whether AP-LoRA can achieve computational efficiency comparable to existing MoE-LoRA frameworks. To address this issue, we adopt two implementation-level optimizations: router parallelization and router–expert fusion.

Specifically, AP-LoRA employs two routers, both of which take account of the input features and expert features. This allows the two routing processes to be merged into a single unified router without altering the underlying model behavior. During training, a major source of inefficiency in MoE-LoRA compared to standard LoRA stems from the introduction of routing. This scheme increases the original two-step LoRA computation into three stages.

To mitigate this issue, AP-LoRA fuses routing weights with the weights of matrix A, enabling the simultaneous computation of intermediate representations and routing weights, thus called "parallel". This fusion significantly increases the computation density and reduces redundant operations, thereby alleviating the efficiency bottleneck commonly observed in MoE-LoRA. After training, expert features are cached and reused during inference, allowing routing decisions to be applied directly without incurring additional computation overhead. We place the pseudocode in Appendix E.

The computational efficiency of AP-LoRA and other MoE-LoRA variants in NLU task is reported

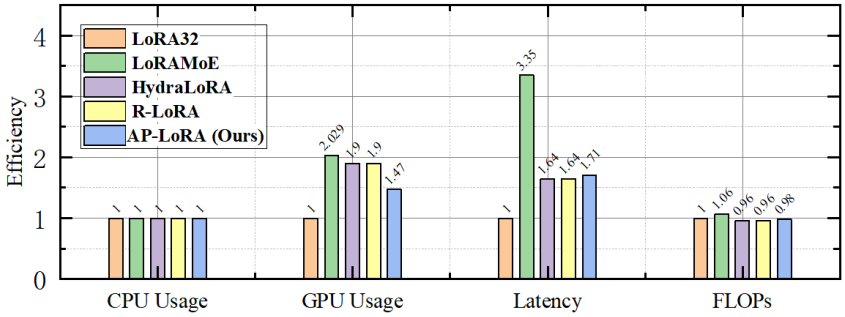

Figure 4: Efficiency analysis across several LoRA with rank 32 and MoE-LoRA variants.

in Figure 4. For each LoRA module, AP-LoRA performs both routing weight computation and expert forward passes, resulting in a computational complexity of

$$\mathcal{O}\big(BLD(n_1+n_2)r + BLD(n_1+n_2)\big),$$

which is slightly lower than that of symmetric MoE-LoRA with an equivalent parameter budget, whose complexity is

$$\mathcal{O}\big(2BLDnr + BLDn\big).$$

B, L and D represent the batch size, sequence length, and input dimonsion, respectively. r denotes the rank of a single expert. n_1, n_2 is the number of matrices A and B.

With router fusion and computation optimizations applied, AP-LoRA achieves an approximately 20% reduction in GPU usage and equal time consumption compared to unoptimized MoE-LoRA variants. The result demonstrates that the increased expressiveness of AP-LoRA does not come at a prohibitive computational cost.

4.6 Ablation Study

MoE architectures. We evaluate three representative MoE-LoRA variants—HydraLoRA, LoRAMoE, and R-LoRA by equipping them with the proposed smart router while keeping their original expert structures unchanged. As reported in Table 3, LoRAMoE (Dou et al., 2024) benefits the most from the smart router, achieving an improvement of 1.66% over its original formulation. HydraLoRA (Tian et al., 2024) and R-LoRA (Liu et al., 2025) also exhibit consistent but smaller gains of 0.07% and 0.22%, respectively.

Nonlinear functions in the router. We further study the impact of different nonlinear transformations in the smart router by replacing the spline function with several fixed alternatives, including

Table 3: MoE-LoRA with Smart Router

Method	Params(%)	GSM8K	gain
LoRAMoE	0.83	50.79	+1.66
HydraLoRA	0.78	51.55	+0.07
rLoRA	0.78	51.17	+0.22
AP-LoRA	0.79	52.46	-

Table 4: Nonlinear Ablations

Function	Params(%)	GSM8K	gain
zero	0.79	50.94	-
tanh	0.79	51.10	+0.16
sigmoid	0.79	50.94	+0
cos	0.79	50.72	-0.22
Spline	0.79	52.46	**+1.52**

tanh, *sigmoid*, and *cos* functions. In addition, we consider a zero function, which reduces the intelligent router to a standard input-only router. The results are summarized in Table 4. Among the fixed nonlinear functions, *tanh* yields a modest improvement of 0.16%, *sigmoid* achieves comparable performance to the zero function, while *cos* results in a slight performance drop of 0.22%. These observations suggest that fixed nonlinear functions are not universally effective for expert–input alignment and, in some cases, may even degrade performance.

5 Conclusion

In this work, we show that MoE-LoRA improves over standard LoRA by composing and optimizing subspaces of low-rank spaces, enabling stronger approximation on heterogeneous tasks. This perspective offers a clear explanation for the effectiveness of AP-LoRA. Extensive experiments on language understanding and arithmetic reasoning benchmarks demonstrate the performance and scalability of AP-LoRA. Our work provides theoretical and practical insights that may inform future MoE-based model adaptation beyond LoRA.

Limitations

While AP-LoRA demonstrates consistent improvements across tasks, two limitations merit acknowledgment. First, as the base model scales up, the performance margin over baselines narrows. We attribute this to the increasing expressiveness of larger pre-trained models, which diminishes the relative contribution of any adapter design. Second, like other MoE-LoRA methods, AP-LoRA does not support weight merging at inference time, incurring additional routing overhead.

References

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. 2021. Training verifiers to solve math word problems. *CoRR*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing*.

Haonan Dong, Wenhao Zhu, Guojie Song, and Liang Wang. 2025. AuroRA: Breaking low-rank bottleneck of LoRA with nonlinear mapping. *arXiv preprint arXiv:2505.18738*.

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. 2024. 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*.

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

Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Yu Han, and Hao Wang. 2024. Mixture-of-LoRAs: An efficient multitask tuning method for large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*.

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*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*.

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*.

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

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. 2023b. LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2024. LoraHub: Efficient cross-task generalization via dynamic LoRA composition. *arXiv preprint arXiv:2307.13269*.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*.

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

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

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

Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. 2024a. When MoE meets LLMs: Parameter efficient fine-tuning for multi-task medical applications. *arXiv preprint arXiv:2310.18339*.

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

Ang Lv, Jin Ma, Yiyuan Ma, and Siyuan Qiao. 2025a. Coupling experts and routers in mixture-of-experts via an auxiliary loss. *arXiv preprint arXiv:2512.23447*.

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

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. PiSSA: principal singular values and singular vectors adaptation of large language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Ashwinee Panda, Vatsal Baherwani, Zain Sarwar, Benjamin Thérien, Supriyo Chakraborty, and Tom Goldstein. 2025. Dense backpropagation improves training for sparse mixture-of-experts. *CoRR*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

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

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

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Tanik Saikh, Tirthankar Ghosal, Amish Mittal, Asif Ekbal, and Pushpak Bhattacharyya. 2022. ScienceQA: A novel resource for question answering on scholarly articles. *International Journal on Digital Libraries*.

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

Mengyang Sun, Yihao Wang, Tao Feng, Dan Zhang, Yifan Zhu, and Jie Tang. 2025. A stronger mixture of low-rank experts for fine-tuning foundation models. *arXiv preprint arXiv:2502.15828*.

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*.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP*.

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

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

Shu Yang, Muhammad Asif Ali, Cheng-Long Wang, Lijie Hu, and Di Wang. 2024. MoRAL: MoE augmented lora for LLMs' lifelong learning. *arXiv preprint arXiv:2402.11260*.

Fangzhao Zhang and Mert Pilanci. 2024. Riemannian preconditioned LoRA for fine-tuning foundation models. In *Proceedings of the 41st International Conference on Machine Learning*.

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

Weilin Zhao, Yuxiang Huang, Xu Han, Zhiyuan Liu, Zhengyan Zhang, Kuai Li, Chen Chen, Tao Yang, and Maosong Sun. 2024. CA-LoRA: Adapting existing LoRA for compressed llms to enable efficient multi-tasking on personal devices. In *First Conference on Language Modeling*.

Heming Zou, Yunliang Zang, Wutong Xu, Yao Zhu, and Xiangyang Ji. 2025. FlyLoRA: Boosting task decoupling and parameter efficiency via implicit rank-wise mixture-of-experts. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

A Proofs of Theoretical Results

A.1 Proof of Lemma 3.1 (Rank Limitation of MoE-LoRA)

Proof. We analyze the rank of MoE-LoRA from two perspectives: the expected rank of the dynamic weight $\mathbb{E}_x[\mathrm{rank}(W(x))]$ and the rank of the expected weight $\mathrm{rank}(\mathbb{E}_x[W(x)])$.

Symmetric MoE-LoRA.

$$\mathbb{E}_x[\mathrm{rank}(W(x))] = \mathbb{E}_x\left[\mathrm{rank}\left(\sum_k \phi_k(x) W_k\right)\right] \leq \sum_k \mathrm{rank}(W_k) \leq Kr,$$

$$\mathrm{rank}(\mathbb{E}_x[W(x)]) = \mathrm{rank}\left(\sum_k \alpha_k W_k\right) \leq Kr,$$

where K is the total number of experts, r is the per-expert rank, and $\alpha_k = \mathbb{E}_x[\phi_k(x)]$ is the expected routing weight for expert k. Under the fixed parameter budget assumption (all experts activated simultaneously), a K-expert MoE-LoRA with per-expert rank r has the same parameter count as a single LoRA of rank Kr. Consequently, the rank bound of Kr is identical to that of a single LoRA of rank Kr, and no rank benefit is gained.

Asymmetric MoE-LoRA. For asymmetric architectures where all B_k share the same A, the matrices $W_k = B_k A$ are row-linearly dependent. Hence,

$$\mathrm{rank}\left(\sum_k \alpha_k W_k\right) = \mathrm{rank}\left(\left(\sum_k \alpha_k B_k\right) A\right) \leq \mathrm{rank}(A) \leq r.$$

Therefore,

$$\mathbb{E}_x[\mathrm{rank}(W(x))] \leq r, \qquad \mathrm{rank}(\mathbb{E}_x[W(x)]) \leq r.$$

From both cases, MoE-LoRA cannot exceed the intrinsic rank restriction of LoRA under a fixed parameter budget, and the asymmetric design can even reduce the effective rank. □

A.2 Proof of Negligibility of $\Delta\Pi_S G$ in Lemma 3.3

Lemma A.1. *Under the approximate orthogonality of expert subspaces (Assumption 3.2), the residual component $\Delta\Pi_S G$ in the decomposition is asymptotically negligible in terms of the nuclear norm.*

Proof. Let $G \in \mathbb{R}^{m\times n}$ be the high-rank target matrix. We define the total low-rank space $S = \mathrm{span}(\bigcup_{k=1}^K S_k)$. Under the assumption of approximate orthogonality, the projection of G onto S can be decomposed as $\Pi_S G = \sum_{k=1}^K \Pi_{S_k} G + \Delta\Pi_S G$, where $\Delta\Pi_S G$ denotes the residual component arising from the imperfect orthogonality of the subspaces.

To establish the negligibility of the residual term $\Delta\Pi_S G$ with respect to the nuclear norm $\|\cdot\|_*$, we first examine the interaction between expert subspaces. Assumption 3.2 implies that the coherence between any two distinct subspaces S_k and S_j is bounded by ϵ, such that the Frobenius inner product of their respective projections satisfies:

$$|\langle \Pi_{S_k} G, \Pi_{S_j} G\rangle_F| \leq \epsilon \|\Pi_{S_k} G\|_F \|\Pi_{S_j} G\|_F.$$

This bound follows from the fact that the operator norm of the product of projection matrices $\|\Pi_{S_k}\Pi_{S_j}\|_2$ is at most ϵ when the underlying bases are ϵ-orthogonal.

By the properties of orthogonal decomposition, the total energy of the projection onto S is given by $\|\Pi_S G\|_F^2 = \|\sum_k \Pi_{S_k} G\|_F^2 + \|\Delta\Pi_S G\|_F^2$. Expanding the first term yields:

$$\left\|\sum_k \Pi_{S_k} G\right\|_F^2 = \sum_k \|\Pi_{S_k} G\|_F^2 + 2\sum_{k<j} \langle \Pi_{S_k} G, \Pi_{S_j} G\rangle_F.$$

Given that the cross-terms are bounded by $\mathcal{O}(\epsilon)$ as shown above, the magnitude of the residual energy is constrained as $\|\Delta\Pi_S G\|_F^2 = \mathcal{O}(\epsilon)$. In the limit of perfect orthogonality ($\epsilon \to 0$), the cross-terms vanish, and the sum of individual projections spans the entirety of $\Pi_S G$, resulting in $\Delta\Pi_S G = 0$.

Finally, we relate this to the nuclear norm via the relationship $\|M\|_* \leq \sqrt{\text{rank}(M)}\|M\|_F$, which is derived from the Cauchy-Schwarz inequality applied to the singular values of M. Since $\Delta\Pi_S G$ is a projection onto a low-rank residual space within S, its rank is bounded by r. Therefore:

$$\|\Delta\Pi_S G\|_* \leq \sqrt{r}\|\Delta\Pi_S G\|_F = \mathcal{O}(\sqrt{r\epsilon}).$$

For a fixed rank r and a negligible constant ϵ, the nuclear norm of the residual component is asymptotically small. Consequently, the approximation error is dominated by the components captured within the expert subspaces, validating the use of the additive low-rank formulation in MoE-LoRA. □

We further provide the detailed derivation for Equation 1 as follows:

$$\begin{aligned}
\mathcal{L} &= \mathbb{E}_x\|G(x) - \sum_k \alpha_k^\star W_k\|_\star - \mathbb{E}_x\|G(x) - \sum_k \phi_k^\star(x) W_k\|_\star, \\
&= \mathbb{E}_x\|(G(x) - \sum_k \phi_k^\star(x) W_k) + \sum_k (\phi_k^\star(x) - \alpha_k^\star) W_k\|_\star - \mathbb{E}_x\|G(x) - \sum_k \phi_k^\star(x) W_k\|_\star, \\
&= \mathbb{E}_x\|\sum_k (\phi_k^\star(x) - \alpha_k^\star) W_k\|_\star.
\end{aligned}$$

Since $\phi_k^\star(x)$ is an optimal router, we get $(G(x) - \sum_k \phi_k^\star(x) W_k) \perp W_k$. The equation holds strictly.

A.3 Proof of Lemma 3.3 (Subspace Approximation)

Proof. Let $G(x) \in \mathbb{R}^{m\times n}$ be an input-dependent target matrix, typically high-rank and thus unattainable by standard LoRA. Let S denote the column space of the standard LoRA update, and S_k the column space of the k-th expert, with $\{S_k\}_{k=1}^K$ being approximately orthogonal (Assumption 3.2). Let W_k be the basis matrix of S_k.

Denote by $\Pi_S G(x)$ the projection of $G(x)$ onto S. Due to approximate orthogonality, there exist optimal coefficients $\{\alpha_k^\star\}$ such that

$$\begin{aligned}
\Pi_S G(x) &= \sum_k \Pi_{S_k} G(x) + \Delta\Pi_S G(x) \\
&= \sum_k \alpha_k^\star W_k + \Delta\Pi_S G(x),
\end{aligned}$$

where $\Delta\Pi_S G(x)$ is the approximation residual from restricting to expert subspaces. Under Assumption 3.2, this term is negligible in nuclear norm; the formal bound is given in Appendix B.

The minimal approximation error of standard LoRA is:

$$\begin{aligned}
Err_{LoRA}^{\min} &= \mathbb{E}_x\|G(x) - \Pi_S G(x)\|_\star \\
&\approx \mathbb{E}_x\left\|G(x) - \sum_k \alpha_k^\star W_k\right\|_\star .
\end{aligned}$$

The improvement of MoE-LoRA over LoRA is then:

$$
\begin{aligned}
\mathcal{L} &= Err^{\min}_{LoRA} - Err^{\min}_{MoE\text{-}LoRA} \\
&= \mathbb{E}_x \left\| G(x) - \sum_k \alpha_k^\star W_k \right\|_\star - \mathbb{E}_x \left\| G(x) - \sum_k \phi_k^\star(x) W_k \right\|_\star \\
&= \mathbb{E}_x \left\| \sum_k (\phi_k^\star(x) - \alpha_k^\star) W_k \right\|_\star \\
&= \sum_k \mathbb{E}_x [|\phi_k^\star(x) - \alpha_k^\star|] \cdot \|W_k\|_\star,
\end{aligned}
$$

where the last step follows from the approximate orthogonality of the expert subspaces and the additivity of the nuclear norm under orthogonality. Since $\mathcal{L} \geq 0$, MoE-LoRA achieves a lower (or equal) minimal approximation error than standard LoRA whenever the routing weights $\phi_k^\star(x)$ exhibit non-zero variance across inputs. Without Assumption 3.2, the result still holds provided Eq. 2 is satisfied, as verified empirically in Appendix B. □

A.4 Proof of Lemma 3.4 (Asymmetric Parallel Architecture)

Proof. Given a target matrix $G(x) \in \mathbb{R}^{m \times n}$, define

$$
G_{\text{pair}} = \sum_k a_k B_k A_k, \qquad G_{\text{mix}} = \left(\sum_j \beta_j B_j \right) \left(\sum_i \alpha_i A_i \right).
$$

Note that $\text{span}\{B_k A_k\}_k \subseteq \text{span}\{B_j A_i\}_{i,j}$, since every product $B_k A_k$ is a special case of $B_j A_i$ with $j = i = k$. The containment is generally strict unless all cross-products $B_j A_i$ ($j \neq i$) happen to lie in $\text{span}\{B_k A_k\}_k$. Therefore,

$$
\begin{aligned}
Err^{\min}_{\text{pair}} &= \mathbb{E}_x \|G(x) - \Pi_{\text{span}\{B_k A_k\}_k} G(x)\|_\star \\
&\geq \mathbb{E}_x \|G(x) - \Pi_{\text{span}\{B_j A_i\}_{i,j}} G(x)\|_\star \\
&= Err^{\min}_{\text{mix}},
\end{aligned}
$$

with strict inequality whenever the enlarged span strictly reduces the projection residual. Under the same parameter budget, the asymmetric parallel architecture therefore achieves a smaller or equal minimal approximation error compared to the symmetric architecture. □

B PCA Observation

To validate that the improvement condition in Eq. (2) holds in practice without relying on Assumption 3.2, we conduct empirical measurements across representative fine-tuning tasks. Specifically, we estimate $\mathbb{E}_x \| \sum_k (\phi_k^\star(x) - \alpha_k^\star) W_k \|_\star$ and $\mathbb{E}_x \| \Delta \Pi_S G(x) \|_\star$ using validation inputs. Across all tasks evaluated, the left-hand side consistently exceeds the right-hand side by a substantial margin, confirming that Lemma 3.3 and Theorem 3.5 hold empirically even when approximate orthogonality is not strictly satisfied.

Table 5: Other Statistic Value: Singular matrices and Weight Variance

Paradiam	**Percentage of singular matrices**	**Variantce of expert weights**
LoRA	32.14	/
MoE-LoRA	33.67	0.093

This content is primarily intended to experimentally validate Equation 2. Since the optimal routing weights ϕ cannot be obtained through quadratic expansion when x is unknown, we instead focus on verifying the validity of the equation in general cases through experiments. Furthermore, we provide supplementary analysis for scenarios that do not satisfy Assumption 3.2.

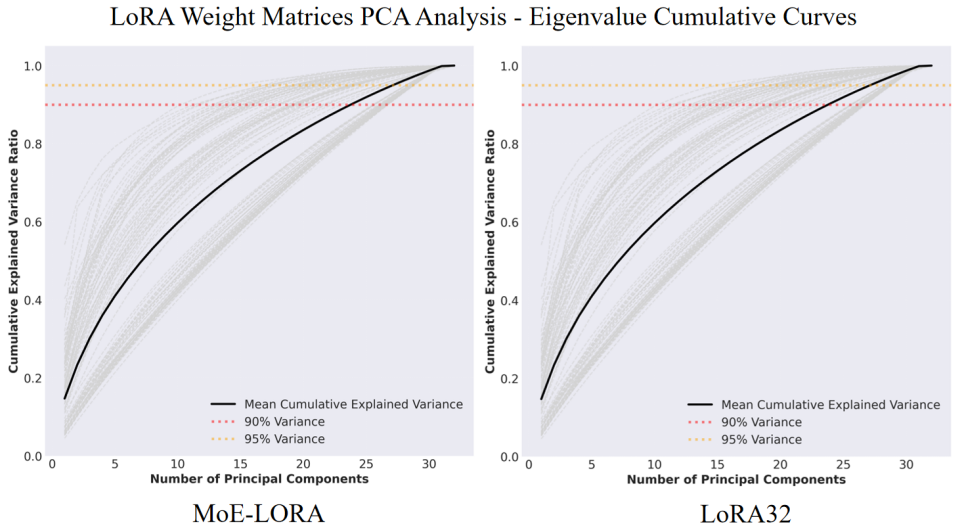

Figure 5: Visualization of Cumulative Explained Ratio Growth with Principle Components.

To accurately measure the magnitude of $\|\Delta\Pi_S G\|_\star$, we perform Principal Component Analysis (PCA) on both the LoRA and MoE-LoRA paradigms for the same task. The matrices of the same layer in MoE-LoRA are merged to match the shape of the LoRA matrix. If the rank of LoRA exceeds that of MoE-LoRA, decomposing LoRA into MoE-LoRA will result in an error represented by $\|\Delta\Pi_S G\|_\star$. Conversely, if the MoE-LoRA matrix is full-rank or shares the same rank as the corresponding LoRA matrix, $\|\Delta\Pi_S G\|_\star$ equals 0.

Table 5 demonstrates that across all matrices, the singular values of MoE-LoRA are slightly larger than those of LoRA. This aligns with our previous conclusion that MoE-LoRA does not expand the rank of LoRA in any way. In MoE-LoRA, there are more than $1/3$ of the singular matrices, yet their ranks are all 31. From the cumulative explained variance shown in Figure 5, we can verify:

$$\frac{\|\Delta\Pi_S G\|_\star}{\sum_k \|W_k\|_\star} \approx 0.01$$

$$\mathbb{E}_x \| \sum_k (\phi_k^\star(x) - \alpha_k^\star) W_k \|_\star > \mathbb{E}_x \|\Delta\Pi_S G(x)\|_\star.$$

Thus, Equation 2 holds, indicating that MoE-LoRA strictly outperforms LoRA on this task.

C Experimental Detail

The experimental detail section serves to ensure the reproducibility of our results. We conducted two primary experiments: **NLU** focuses on natural language understanding, while **AR** investigates the arithmatic reasoning capability. All experiments were implemented using PyTorch.

Table 6 summarizes the specific configurations and hyperparameters used in both experiments.

D Dataset Details

In this section, we provide a comprehensive description of the datasets used in our experiments, categorized by task.

D.1 Natural Language Understanding (NLU)

Our NLU evaluation covers a diverse set of tasks including linguistic competence, semantic similarity, and question answering.

GLUE Benchmark We select three representative tasks from the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018):

Table 6: Hyperparameter settings and hardware configurations for Experiment A and B.

Hyperparameter	**NLU**	**AR**
Base Model	Qwen-3-0.6B/Llama-7B	Llama-3.2-3B
LoRA Rank (r)	8/32	32
Per Expert Rank (r)	8	8
AP-LoRA Experts ($n1, n2$)	(3, 4)	(3, 4)
n in Spline Function	5	5
Cut-off	256	256
Learning Rate	3×10^{-4}	3×10^{-4}
Batch Size	8	4
Optimizer	AdamW	AdamW
Training Epochs	10/20	3
Precision	BF16	BF16
Hardware	NVIDIA RTX 4090D/NVIDIA A800	NVIDIA RTX 4090D

- **CoLA**: The Corpus of Linguistic Acceptability, a single-sentence task with 8.55k training examples for evaluating grammatical acceptability.
- **MRPC**: The Microsoft Research Paraphrase Corpus, a sentence-pair task with 3.67k training examples for paraphrase detection.
- **RTE**: Recognizing Textual Entailment, an inference task containing 2.49k training examples.

Question Answering (QA) To evaluate broader reasoning, we include three specialized QA datasets:

- **CommonsenseQA (CSQA)**: Focuses on complex commonsense reasoning.
- **ScienceQA**: A multi-modal science question dataset.
- **OpenbookQA (OBQA)**: Modeled after Openbook exams to test deep external knowledge retrieval.

D.2 Arithmetic Reasoning

To assess multi-step mathematical derivation, we utilize the following datasets:

- **Math10K**: A comprehensive arithmetic dataset from *LLM-Adapter*, used as our primary training base.
- **Arithmetic Benchmarks**: We evaluate on *GSM8K*, *AQuA*, *SVAMP*, and specific subsets: **MultiArith** (multi-step arithmetic), **AddSub** (addition/subtraction), and **SingleEQ** (single-equation problems).

Table 7: Detailed task descriptions and dataset statistics. GLUE task statistics are derived from the benchmark summary.

Corpus	Task	Metrics	# Labels	# Train	Domain
		Natural Language Understanding (NLU)			
CoLA	Acceptability	Matthews Corr.	2	8.55k	misc.
MRPC	Paraphrase	Accuracy/F1	2	3.67k	News
RTE	NLI	Accuracy	2	2.49k	News & Wikipedia
CSQA	QA	Accuracy	5	9.7k	Commonsense
ScienceQA	QA	Accuracy	3	6.5k	Science
OBQA	QA	Accuracy	4	4.9k	Science
		Arithmetic Reasoning			
Math10K	Math	Accuracy	-	10k	Arithmetic
GSM8K	Math	Accuracy	-	7.47k	Grade School Math
MultiArith	Math	Accuracy	-	600	Arithmetic
SingleEQ	Math	Accuracy	-	508	Arithmetic
AQuA	Algebra	Accuracy	5	254	Algebra
SVAMP	Math	Accuracy	-	1k	Arithmetic
AddSub	Math	Accuracy	-	395	Arithmetic

E Fast Algorithm for AP-LoRA

We provide the pseudocode for the efficient computation of AP-LoRA. The algorithm obtains routing logits and expert intermediate representations via a single fused matrix multiplication, and reuses cached expert features during inference to achieve high computation density with minimal overhead.

Algorithm 1 Fast Computation of AP-LoRA via Fused Projection

Require: Input hidden state $x \in \mathbb{R}^d$; Fused projection matrix $W \in \mathbb{R}^{(n_1+n_2+n_1 r)\times d}$; Up-projection LoRA matrix $B \in \mathbb{R}^{d\times r}$.

Ensure: Output feature $y \in \mathbb{R}^d$.

// Perform a single GEMM to obtain all intermediate routing and projection signals

1: $z \leftarrow Wx \in \mathbb{R}^{n_1+n_2+n_1 r}$

// Decompose the fused vector z into router logits and the LoRA base

2: Split z into components:

- $z^{(1)} \leftarrow z[1 : n_1]$ // Logits for the first routing stage
- $z^{(2)} \leftarrow z[n_1 + 1 : n_1 + n_2]$ // Logits for the second routing stage
- $H \leftarrow \text{reshape}(z[n_1 + n_2 + 1 : \text{end}], (n_1, r))$ // Low-rank base matrices for n_1 experts

// Compute the weighted hidden representation for the first router

3: $\tilde{h} \leftarrow \sum_{i=1}^{n_1} z_i^{(1)} \cdot H_i \in \mathbb{R}^r$

where H_i is the i-th row of H

// Map the low-rank representation back to the model dimension

4: $y_{\text{mid}} \leftarrow B\tilde{h} \in \mathbb{R}^d$

// Apply the second-stage routing weights to produce the final output

5: $y \leftarrow \left(\sum_{j=1}^{n_2} z_j^{(2)}\right) \cdot y_{\text{mid}}$

6: **return** y