Learning to Select, Not Relearn: Hard-Routed Mixtures of Reasoning LoRAs

Anonymous ACL submission

Abstract

Composing independently trained LoRA experts into a single large language model is a practical approach for multi-domain adaptation, particularly when original training data cannot be shared. However, existing mixture-of-LoRA methods rely on soft routing, introducing a mismatch with the original LoRA assumption and often requiring additional retraining of experts. Separately, conventional supervised fine-tuning encourages short-form answer imitation and provides limited supervision for multi-step reasoning in instruction-tuned models. We propose **Hard-Routed MoR-LoRA**, a framework that decouples reasoning acquisition from expert selection. First, domain-specific LoRA adapters are trained independently using reinforcement learning from verifiable feedback to obtain reasoning experts. We then distill reasoning traces from these experts and train only a lightweight shared router and a small attention adaptation while keeping all experts frozen. Integration uses hard top-1 routing with a straight-through estimator, applying exactly one expert per token with unit scaling. Experiments across five benchmarks and multiple model scales show that our method preserves reasoning performance while using substantially fewer trainable parameters and consistently outperforms soft-routing mixture baselines.

1 Introduction

Large language models (LLMs) have demonstrated strong performance across diverse language understanding and reasoning tasks (Brown et al., 2020). However, adapting these models to specialized domains remains expensive because full fine-tuning requires substantial computation and access to training data. Parameter-efficient fine-tuning (PEFT) methods address this issue by modifying only a small subset of parameters (Houlsby et al., 2019). Among them, Low-Rank Adaptation (LoRA) (Hu et al., 2022) is widely used because it adds lightweight trainable matrices while keeping the backbone frozen.

In practice, domain adaptation is often done independently, where different groups train LoRA adapters on their own datasets but cannot share the original data due to privacy or regulatory constraints. Instead, they may release only the trained LoRA modules. A natural goal is therefore to compose multiple independently trained LoRA experts into a single model that supports heterogeneous tasks. Mixture-of-Experts (MoE) architectures (Shazeer et al., 2017; Fedus et al., 2022) provide a natural mechanism for such modular reuse by routing each token to selected experts. However, directly applying standard MoE routing to LoRA introduces a fundamental mismatch. LoRA modules are trained assuming a unit additive update, whereas MoE combines experts using routing weights. Soft routing, therefore, blends expert behaviors and changes their function, requiring additional fine-tuning of the experts after integration. Prior approaches, such as LoRA-Mixer(Li et al., 2025a), address this by retraining the adapters with auxiliary losses, but this increases trainable parameters, requires more data, and can alter the knowledge encoded in the original experts.

This challenge becomes more pronounced for instruction-tuned models with chain-of-thought (CoT) reasoning ability. Conventional fine-tuning encourages imitation of short answers and may weaken internal reasoning behavior in large models, while reinforcement learning from verifiable feedback (RLVF) can produce stronger reasoning experts by optimizing correctness rather than matching reference outputs (Shao et al., 2024; Guo et al., 2025). However, existing LoRA composition methods primarily focus on supervised multi-task fine-tuning and do not study the training and integration of experts in instruction-tuned models that exhibit CoT reasoning.

In this work, we propose **Hard-Routed MoR-LoRA**, a framework that separates reasoning acquisition from expert selection. First, domain-specific LoRA adapters are trained independently using RLVF to obtain reasoning. Next, the experts are frozen and integrated using a hard-routed mixture that selects exactly one expert per token through a straight-through estimator (STE). We distill reasoning traces from the experts and train only a shared lightweight router and a small attention adaptation using a small amount of data (1000 samples per dataset). Hard routing preserves the original LoRA assumption by applying each expert with unit scale, enabling modular composition without retraining the experts.

Experiments across five benchmarks and multiple model scales show that RLVF produces stronger reasoning experts for capable models, while conventional fine-tuning can degrade reasoning behavior in large instruction-tuned models. We further demonstrate that hard routing achieves better performance than soft routing while using substantially fewer trainable parameters. Analysis of routing probabilities reveals that normalized soft top-k mixtures implicitly behave like near hard top-1 selection, indicating that hard routing captures the intended behavior more directly.

Our contributions are summarized as follows:

- We identify a scaling mismatch between LoRA modules and soft MoE routing and propose a hard-routing formulation that preserves unit-scale expert application without retraining experts.
- We introduce a two-stage framework that trains reasoning LoRA experts using RLVF and integrates them through distillation while keeping experts frozen.
- We provide empirical evaluation across model scales and tasks showing improved performance and parameter efficiency, along with an analysis explaining the behavior of soft routing.

2 Related Work

Modular LoRA adaptation. LoRA (Hu et al., 2022) enables efficient fine-tuning by learning small low-rank updates while keeping the backbone model frozen. Although effective for adapting a model to a single task, independently trained LoRA modules are usually task-specific and difficult to reuse across different domains, especially when the original training data cannot be shared. Several works, therefore, study how to combine multiple pretrained adapters into one model. LoraHub (Huang et al., 2023) learns weights that merge a set of LoRA modules into a single composed adapter, while LoRA-LEGO (Zhao et al., 2024) decomposes adapters into smaller rank components and reconstructs a merged adapter for cross-task reuse. These methods demonstrate that pretrained adapters can be combined without retraining, but the resulting model applies the same merged adapter to every input and cannot dynamically select different experts at the token level.

Mixture-of-Experts with LoRA. MoE architectures (Shazeer et al., 2017; Fedus et al., 2022) increase model capacity by routing tokens to a sparse subset of experts. Instead of merging adapters into one module, MoE-based approaches keep multiple experts and select among them during inference. Recent work applies this idea to parameter-efficient tuning by treating LoRA modules as experts. MixLoRA (Li et al., 2024) inserts multiple LoRA modules into feed-forward layers (FFN) with top-k routing. MoLE (Wu et al., 2024) learns hierarchical gating across layer-wise adapters, while SiRA (Zhu et al., 2023) and GOAT (Li et al., 2025b) study alignment and sparse routing strategies. The closest to our work, LoRA-Mixer (Li et al., 2025a), integrates LoRA modules into attention projections and trains a lightweight router. However, soft routing produces a weighted sum of LoRA updates, whereas standalone LoRA training assumes unit-scale application. To address this mismatch, LoRA-Mixer further fine-tunes the experts with regularization losses, which increases the number of trainable parameters and requires additional data and computation.

Instruction-tuned Reasoning and Verifiable Feedback. Instruction tuning and alignment stages shape how language models produce and structure reasoning (Ouyang et al., 2022). RLVF improves reasoning by optimizing correctness using automatically checkable rewards instead of matching reference outputs (Shao et al., 2024; Guo et al., 2025). DeepSeek-R1 (Guo et al., 2025) shows that strong reasoning ability can emerge from this training paradigm, and Group Relative Policy Optimization (GRPO) (Shao et al., 2024) stabilizes training without requiring a learned value critic. In our setting, adapters are added to instruction-tuned base models that already ex-

hibit CoT behavior. Preserving this behavior during training is important, and RLVF provides a natural way to train reasoning experts without overriding the aligned reasoning patterns of the base model.

Hard routing and Straight-through Estimators. Hard routing selects a single expert through a discrete decision, which is not directly differentiable. The STE enables gradient-based training by using the hard decision in the forward pass and a surrogate gradient in the backward pass (Bengio et al., 2013; Liu et al., 2023). In our approach, STE allows top-1 expert selection with unit scaling, preserving the original LoRA formulation while still allowing end-to-end training of the router.

3 Preliminaries

This section briefly introduces the components utilized in our method: Low-Rank Adaptation, Mixture-of-Experts, and Reinforcement Learning from Verifiable Feedback.

3.1 Low-Rank Adaptation

LoRA (Hu et al., 2022) parameterizes weight updates as a low-rank decomposition:

$$W_{\text{updated}} = W_{\text{pretrained}} + AB \tag{1}$$

where $A \in \mathbb{R}^{n\times r}$ and $B \in \mathbb{R}^{r\times n}$ with $r \ll n$. During fine-tuning, only A and B are optimized while $W_{\text{pretrained}}$ remains frozen, substantially reducing memory and computational cost.

3.2 Mixture-of-Experts Routing

Given token representation x, a router produces logits $\mathbb{G}(x) \in \mathbb{R}^K$ and probabilities

$$\mathbb{P}_i(x) = \text{softmax}(\mathbb{G}(x))_i \tag{2}$$

Standard MoE combines multiple experts using routing weights (Fedus et al., 2022), typically activating the top-k experts and scaling each expert output by its routing probability. The final output is computed as

$$\text{MoE}(x) = \sum_{i=1}^{K} \mathbb{I}(i \in \text{top-}k(\mathbb{P}(x))) \cdot \mathbb{P}_i(x) \cdot E_i(x) \tag{3}$$

where $\mathbb{I}(\cdot)$ denotes the indicator function and $E_i(x)$ is the i-th expert's output.

3.3 Reinforcement Learning from Verifiable Feedback

RLVF trains models using rewards derived from automatically verifiable outputs rather than human preference annotations (Shao et al., 2024; Guo et al., 2025). Unlike supervised fine-tuning, the model is not constrained to follow a reference reasoning trace; only the final answer is evaluated by the verifier. This allows the model to freely generate intermediate CoT reasoning steps, encouraging the emergence of structured reasoning behavior.

We adopt GRPO (Shao et al., 2024). For a prompt x, a group of reasoning trajectories $\{\tau_i\}_{i=1}^N$ is sampled with rewards $\{r_i\}_{i=1}^N$:

$$\bar{r} = \frac{1}{N}\sum_{i=1}^{N} r_i \tag{4}$$

$$A_i = r_i - \bar{r} \tag{5}$$

where A_i is the advantage for the i-th reasoning trace.

To improve sample efficiency, we use traces collected under the behavior policy π_{old}, which is a few checkpoints behind the target policy π_θ. For each trajectory τ_i, we compute a token-level importance ratio:

$$w_{i,t} = \frac{\pi_\theta(y_{i,t} \mid x, y_{i,<t})}{\pi_{\text{old}}(y_{i,t} \mid x, y_{i,<t})} \tag{6}$$

The off-policy token-level GRPO objective becomes:

$$\mathcal{L}_{\text{off}}(\theta) = -\mathbb{E}_x\left[\frac{1}{N}\sum_{i=1}^{N} A_i \sum_{t=1}^{T_i} w_{i,t} \log \pi_\theta(y_{i,t} \mid x, y_{i,<t})\right] \tag{7}$$

4 Methodology

We propose **Hard-Routed MoR-LoRA (Mixture of Reasoning LoRA)**, a plug-in framework that integrates multiple reasoning LoRA experts into a unified model while preserving the original LoRA assumption and maintaining sparse computation. The key idea is to decouple *reasoning acquisition* from *expert selection*. First, each LoRA adapter independently learns domain-specific reasoning behavior using RLVF (Shao et al., 2024; Guo et al., 2025). Then, the pretrained experts are combined through a hard-routed mixture in which only a lightweight router and a small attention adaptation are trained using supervised fine-tuning (SFT) on distilled reasoning traces.

The overall two-stage pipeline is illustrated in Figure 1. Stage I trains reasoning experts independently, while Stage II integrates them into a shared model by learning which expert should be selected for each token.

4.1 Hard Mixture-of-LoRAs Architecture

Mismatch in Soft Mixture-of-LoRAs

From Eq. 1, a pretrained LoRA expert is applied with unit contribution. However, the standard MoE applies weighted combinations of multiple experts:

$$W_{\text{updated}} = W_{\text{pretrained}} + \sum_{i=1}^{k} \omega_i A_i B_i \tag{8}$$

where ω_i are routing probabilities.

This changes the magnitude of the update. Since each LoRA expert was trained assuming unit contribution, scaling by ω_i alters its effective function and introduces a distribution shift between standalone training and mixture usage. Prior work compensates for this mismatch by further fine-tuning experts by introducing preservation losses (Li et al., 2025a) to prevent a drastic shift in the already trained LoRA adapters.

Hard Routing with Straight-Through Estimator

To preserve the original LoRA assumption, the selected expert must be applied with unit scale. However, a softmax router produces continuous weights. In a top-1 setting, enforcing unit contribution by normalizing a single routing weight is a constant function with zero gradient. Therefore, a purely top-1 soft routing formulation cannot simultaneously guarantee unit scaling and allow gradient-based optimization.

Instead, we enforce *top-1 routing* via discrete expert selection. For a token x:

$$\mathbb{P} = \text{softmax}(\mathbb{G}(x)) \tag{9}$$

$$j = \arg\max_i (\mathbb{P}_i) \tag{10}$$

This produces a one-hot routing decision. However, the $\arg\max$ operation is non-differentiable, preventing backpropagation through the router. To enable training, we use the STE (Liu et al., 2023), which keeps the hard decision in the forward pass but approximates gradients using the soft probabilities in the backward pass:

$$Y_{\text{hard}} = \text{one-hot}(j) \tag{11}$$

$$Y = \text{SG}(Y_{\text{hard}} - \mathbb{P}) + \mathbb{P} \tag{12}$$

where $\text{SG}(\cdot)$ stops gradients. The forward pass uses the discrete routing vector Y_{hard}, while gradients propagate through P. The final output becomes

$$\text{STE}(x) = \sum_{i=1}^{K} Y_i E_i(x) = E_j(x) \tag{13}$$

which guarantees unit scaling of the selected LoRA expert while still allowing the router to be optimized via gradient descent.

Parameter Efficiency. All pretrained experts remain frozen. A single linear router is shared across layers, and only a small LoRA module on attention layers is trainable during integration. The number of trainable parameters, therefore, does not grow with the number of experts while maintaining sparse computation.

4.2 Training

The training is done in two stages:
Stage I: Train domain-specific LoRA experts using RLVF (Shao et al., 2024; Guo et al., 2025).
Stage II: Freeze experts and train the router using SFT on distilled traces.

This separation preserves reasoning capability while enabling efficient modular composition.

Stage I: Training Reasoning LoRA Experts

Each dataset trains one LoRA adapter independently while the base model remains frozen. No router is present in this stage. The adapters are optimized using RLVF with GRPO so that each expert specializes in domain-specific reasoning and produces CoT traces. After training, we obtain a set of reasoning experts: $\{E_1, E_2, \ldots, E_K\}$

Stage II: Mixture Integration via Distillation

Distillation Data Generation. We query each trained expert with prompts from its domain and collect generated reasoning traces. These traces form a supervised dataset representing expert behavior. A limited number of samples is sufficient to construct the SFT dataset, since the objective of this stage is to learn the routing policy rather than to relearn reasoning ability. The reasoning knowledge is already encoded in the frozen experts, and therefore, the training data only needs to cover representative input patterns for correct expert selection.

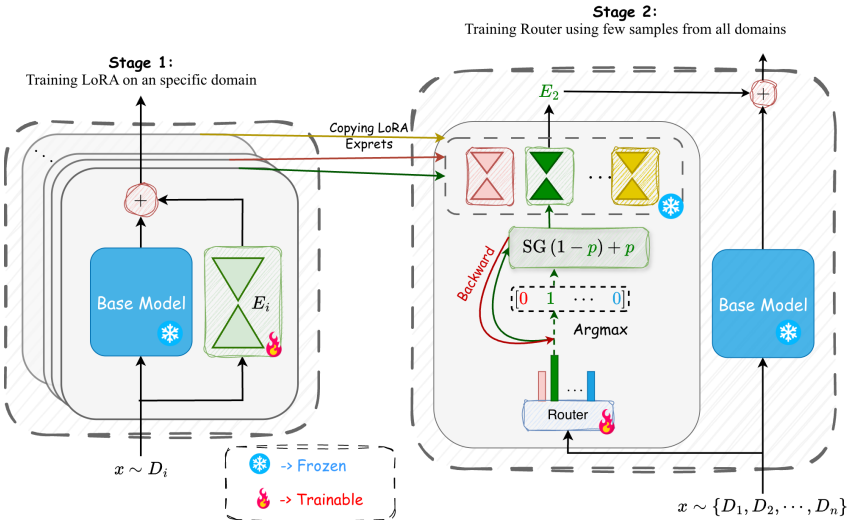

Figure 1: **Overview of the proposed Hard-Routed MoR-LoRA framework.** The method follows a two-stage training pipeline. **Stage I (left):** Each LoRA adapter is independently trained on a specific domain using reinforcement learning from verifiable feedback while the base model remains frozen, producing a set of reasoning experts. **Stage II (right):** All pretrained experts are integrated into a shared model and kept frozen. A shared lightweight router and a small attention LoRA are trained using supervised fine-tuning on distilled reasoning traces collected from the experts. During inference, the **router** selects a single expert per token via $\arg\max$, enforcing unit scaling and preserving the original LoRA assumption. STE enables gradients to propagate through the discrete routing decision by following the soft probabilities during the backward pass (red arrow).

Router and Attention Adaptation. All experts remain frozen. We train **a shared linear router**, and **LoRA adapters on attention layers** using standard next-token likelihood on the distilled traces. The objective is to learn expert selection rather than relearn reasoning ability.

5 Experiments

We evaluate Hard-Routed MoR-LoRA along two independent dimensions: (1) the quality of reasoning experts obtained via RLVF, and (2) the effectiveness of hard routing compared to soft mixture routing during expert integration.

Experimental Setup

We conduct experiments on instruction-following language model Meta-LLaMA-3 (Dubey et al., 2024) at three model scales, 1B, 3B, and 8B[1] parameters. We also evaluate additional model families in Appendix C.

Datasets. To evaluate performance across heterogeneous domains, we consider five representative tasks: (a) mathematical reasoning, (b) commonsense reasoning, (c) medical multiple-choice question answering, (d) reading comprehension, and (e) grammatical understanding.

These tasks are evaluated using the GSM8K (Cobbe et al., 2021), ARC-Challenge (ARC-C) (Clark et al., 2018), Medical Question Answering (MedQA) (Jin et al., 2021), BoolQ (Clark et al., 2019), and Corpus of Linguistic Acceptability (CoLA) (Warstadt et al., 2019) datasets, respectively. Each dataset covers a distinct domain with little to no overlap.

Compared Models. We compare against the LoRAMixer (Li et al., 2025a) approach, which was the state-of-the-art method to integrate non-reasoning LoRA adapters into one unified model.

- **Independent SFT:** standalone fine-tuning dataset-specific adapters on the FFN layers.
- **Independent RLVF:** standalone reasoning dataset-specific on the FFN layers.
- **LoRAMixer TopK=1:** soft routing with continued adapter training using the preservation loss.
- **LoRAMixer TopK=2 Normalized:** soft routing with renormalized weights and continued adapter training.
- **Ours (Hard-Routed MoR-LoRA):** hard routing via STE with frozen experts with LoRA fine-tuning on the attention layer.

[1] `meta-llama/Llama-3.2-1B-Instruct`, `meta-llama/Llama-3.2-3B-Instruct`, `meta-llama/Llama-3.1-8B-Instruct`.

Following the LoRAMixer training protocol, adapters are further optimized during mixture training with an additive L2 preservation loss to limit deviation from their pretrained parameters. In contrast, our method keeps all experts frozen and trains only the routing module and a lightweight attention adaptation. Additionally, during mixture training, only 1000 samples per dataset are used.

The independent versions train adapters tailored to a single dataset. Then, in the second stage of the training, they are integrated into one unified model.

5.1 RLVF Produces Stronger Reasoning Experts

We first evaluate the quality of individual experts trained in Stage I before any mixture integration. This isolates the effect of the training paradigm. Figure 2 summarizes the average accuracy across model scales, while full per-task results are provided in Appendix A.1, Table 3.

For medium and large models (3B and 8B), RLVF substantially improves performance on reasoning-intensive tasks. For example, the 3B model improves from 68.60 (SFT) to 72.69 with RLVF, and the 8B model improves from 72.72 to 79.72. Unlike fine-tuning, which encourages imitation of reference outputs, RLVF allows the model to explore intermediate reasoning steps and optimize correctness, leading to stronger multi-step reasoning behavior.

The results also show that ordinary fine-tuning can degrade performance in large instruction-tuned models. We attribute this to fine-tuning, overriding the model's inherent COT behavior by encouraging short-form answer imitation. In contrast, RLVF preserves the aligned reasoning behavior because optimization depends only on verifiable correctness rather than matching a specific reasoning trace. For the 1B model, on the other hand, RLVF provides smaller gains compared to fine-tuning (32.61 vs 49.18). This is likely due to insufficient reasoning capacity as the 1B model struggles to produce diverse and correct reasoning trajectories, weakening the reward signal. Thus, the benefit of RLVF increases with model capability.

As detailed in Table 3, the improvement primarily appears in reasoning-heavy datasets such as GSM8K and ARC-C, while simpler linguistic tasks such as CoLA show limited advantage from RLVF. This supports the view that RLVF primarily strengthens reasoning behavior rather than general language modeling.

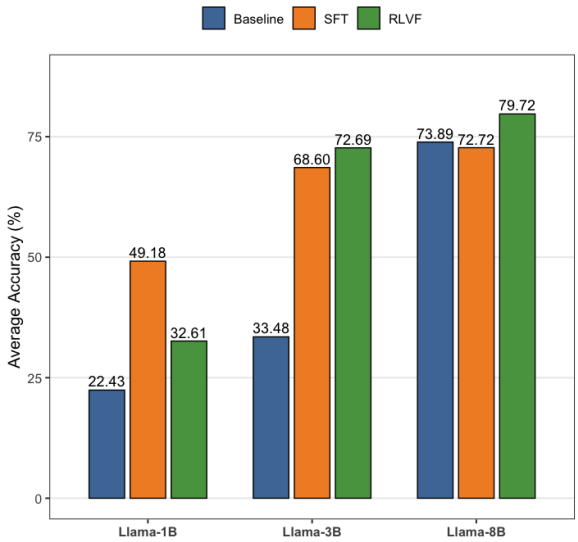

Figure 2: **Performance of standalone SFT and RLVF.** RLVF improves performance for 3B and 8B models but not for 1B, suggesting that reasoning-oriented training benefits models with sufficient capacity while preserving CoT behavior.

5.2 Hard Routing Improves Expert Integration

We now evaluate Stage II, where pretrained experts are integrated using only 1000 samples per dataset. Figure 3 summarizes the integration results for both SFT experts and RLVF experts, and Appendix A.2, Tables 4 and 5 report the full numbers across five tasks and trainable parameter counts.

Hard-Routed MoR-LoRA achieves better performance while requiring substantially fewer trainable parameters. Unlike LoRAMixer, which retrains the already-trained experts to compensate for the scaling mismatch introduced by soft routing, our architecture preserves the original LoRA formulation and therefore only learns expert selection while keeping the experts frozen. This shows that the main gains come from correct routing rather than modifying expert knowledge. For example, on the 3B with RLVF experts, our method reaches 72.07 average while training ≈73M parameters, whereas LoRAMixer Top-1 reaches 65.67 while training ≈606M parameters.

As shown in Figure 3, the improvement from hard routing over soft routing is larger for the 1B and 3B models compared to the 8B model. From Stage I results, these smaller models benefit more from trained experts relative to the base model, so preserving expert behavior during integration is more important. Therefore, correcting the LoRA

integration mismatch through hard routing has a stronger effect. For the 8B model, the relative improvement is smaller but still consistent, with a particularly large gain on GSM8K as shown in Appendix Table 5, where the reasoning expert provides a strong improvement over the base model after RLVF training and hard routing better maintains its effectiveness. We also compare our approach with a static adapter integration method, LoRAHub, in Appendix D.

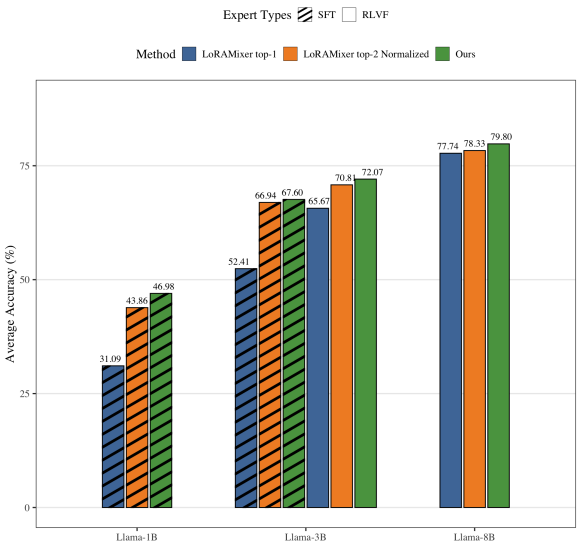

Figure 3: **Hard routing vs. soft routing for LoRA composition.** Across model scales and expert types (SFT and RLVF), Hard-Routed MoR-LoRA achieves equal or better performance than soft routing despite not retraining experts, indicating that routing quality dominates expert adaptation.

5.3 Soft Routing Collapses Toward Single-Expert Selection

Hard routing resolves the LoRA integration mismatch by ensuring that exactly one expert with unit scaling is applied. In contrast, soft routing combines multiple experts using routing weights. While top-1 soft routing cannot maintain unit scaling, top-2 routing can be normalized so that the total update magnitude remains constant.

As shown in Table 4 and Table 5, top-2 normalized routing performs substantially better than top-1 soft routing and approaches the performance of hard routing. However, examining the routing behavior reveals that the two experts are not used equally.

Figure 4 shows the distribution of the dominant routing weight (the larger of the two selected weights). The router consistently assigns most of the probability mass to a single expert, with an average dominant weight of approximately 0.71. In many cases, one expert receives more than twice the weight of the other. Therefore, the top-k normalized soft-router implicitly behaves like a near top-1 selector. Nonetheless, the model still evaluates and combines multiple experts, producing an update that does not correspond to any standalone LoRA module and introducing *unnecessary computation*. Hard routing removes this redundancy by directly selecting a single expert, preserving sparsity and consistency with the LoRA formulation while achieving comparable or better performance.

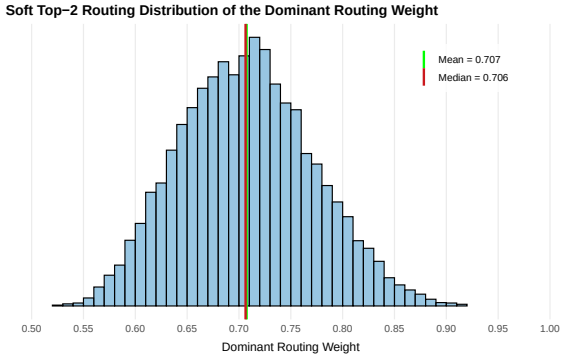

Figure 4: **Distribution of the dominant routing weight under top-2 normalized routing.** The router assigns most probability mass to one expert, indicating that the model implicitly performs near top-1 selection. Hard routing makes this behavior explicit while preserving sparsity and avoiding mismatched expert combinations.

5.4 Ablation Study

We perform ablations to determine whether Stage II improves performance through additional learning or mainly through better expert selection, with further analysis of the LoRA rank provided in Appendix B.

Is Reinforcement Learning Necessary for Integration?

Stage II of our method aims to learn expert selection rather than relearning reasoning behavior. During this stage, all experts remain frozen, and only the shared router and attention LoRA are optimized. We compare two training strategies: (1) SFT on distilled reasoning traces generated by the experts (our default setting), and (2) RLVF.

Results on the 3B model are shown in Table 1. RLVF achieves slightly higher performance (+0.93 average), but requires substantially higher computation due to trajectory sampling and advantage eval-

uation. The small improvement indicates that Stage II primary learns which expert to select, while the reasoning capability is already encoded in the pretrained experts. Therefore, SFT on distilled traces is sufficient for integration, and reinforcement learning is primarily necessary only during expert training.

Method	GSM8K	ARC-C	MedQA	BoolQ	CoLA	Avg
RLVF	75.01	75.40	52.00	85.99	76.58	**73.00**
Distill	73.62	75.43	51.53	83.09	76.70	72.07

Table 1: Comparison of training strategies for Stage II integration on the 3B model. Both methods train only the router and attention LoRA while keeping experts frozen and using the same prompts.

Sample Training Budget

We analyze how many supervision samples are required for Stage II. For each dataset, we randomly sample prompts and generate distilled reasoning traces from the pretrained experts. The mixture model is then trained for one epoch on the combined distilled data from all five domains. We compare using 750 and 1000 prompts per dataset against using all available samples (30,573).

Table 2 shows that using only 1000 prompts per dataset achieves performance close to training on the full dataset, with less than 1% difference in average accuracy. Increasing the amount of data yields only marginal gains, indicating that Stage II mainly learns routing patterns rather than domain knowledge. Since reasoning ability is already encoded in the experts trained in Stage I, only a small number of examples is sufficient to identify the appropriate expert.

# Samples	GSM8K	ARC-C	MedQA	BoolQ	CoLA	Avg
5×750	70.36	73.72	52.57	79.88	74.45	70.20
5×1000	73.62	75.43	51.53	83.09	76.70	72.07
ALL (30,573)	75.32	75.02	52.32	85.12	77.10	**72.98**

Table 2: Effect of the Stage II training budget on the 3B model. The first two rows use the specified number of prompts per dataset (five datasets total).

6 Conclusion

We study how independently trained LoRA adapters can be combined in instruction-tuned language models without changing their behavior. We find that the main challenge in pretrained LoRA composition comes from the mismatch between unit-scale LoRA updates and weighted mixture routing, which requires prior methods to retrain experts and increase parameters. Hard-Routed MoR-LoRA instead treats integration mainly as a routing problem: reasoning experts are first trained independently using RLVF, and the mixture is then trained only to select among them while keeping all experts frozen. Across multiple model scales and tasks, we observe three consistent results. First, verifiable-feedback training produces stronger reasoning experts for capable instruction-tuned models. Second, hard routing integrates these experts more reliably and with far fewer trainable parameters than soft routing approaches. Third, normalized soft mixtures tend to behave like near single-expert selection, explaining why expert retraining is unnecessary. Overall, our results show that modular adaptation can be achieved by learning when to use each expert rather than modifying what each expert knows, enabling practical reuse of domain experts under limited data or data-sharing constraints.

7 Limitations

Although our approach avoids retraining experts, integration is not completely training-free. Stage II still requires a small amount of labeled data to learn the routing behavior. As a result, adding new experts or domains requires additional routing data and training, and the framework does not yet support true zero-shot integration. This limitation is more significant in settings where only very few or no samples are available.

Our method also assumes that each prompt mainly belongs to a single domain at inference time. Experts are trained on separate datasets, and the router selects one per token. While this matches many practical scenarios with clearly separated tasks, real prompts may require multiple types of knowledge at the same time. Furthermore, if a prompt lies outside the domains covered by the available experts, the router must still choose the closest expert, which may cause systematic errors instead of gradual performance degradation. Evaluating routing-based adapter composition under cross-domain and out-of-domain inputs would require datasets containing mixed-domain prompts, which we leave for future work.

References

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or Propagating Gradients through Stochastic Neurons for Conditional Computation. *arXiv preprint arXiv:1308.3432*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language Models are Few-shot Learners. *Advances in neural information processing systems*, 33:1877–1901.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BooQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. *arXiv preprint arXiv:1905.10044*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 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*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*.

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*, 23(120):1–39.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, and Ramona Merhej. 2025. Gemma 3 Technical Report. *arXiv preprint arXiv:2503.19786*.

Daya Guo, Dejian Yang, Haowei Zhang, and 1 others. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in Large Language Models via Reinforcement Learning. *arXiv preprint arXiv:2501.12948*.

Neil Houlsby, Sebastian Jastrzebski, Andrzej Brooks, Rosanne de Vries, Andrea Guedj, and Grégory Nematzadeh. 2019. Parameter-Efficient Transfer Learning for NLP. In *International Conference on Machine Learning (ICML)*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. LoRA: Low-Rank Adaptation of Large Language Models. *ICLR*, 1(2):3.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2023. LoRAHub: Efficient Cross-Task Generalization via Dynamic LoRA Composition. *arXiv preprint arXiv:2307.13269*.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. What Disease Does This Patient Have? A Large-scale Open Domain Question Answering Dataset From Medical Exams. *Applied Sciences*, 11(14):6421.

Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan Zhang, Lei Duan, Jie Zuo, Cal Yang, and 1 others. 2024. MixLoRA: Enhancing Large Language Models Fine-tuning with LoRA-based Mixture of Experts. *arXiv preprint arXiv:2404.15159*.

Wenbing Li, Zikai Song, Hang Zhou, Yunyao Zhang, Junqing Yu, and Wei Yang. 2025a. LoRA-Mixer: Coordinate Modular LoRA Experts Through Serial Attention Routing. *arXiv preprint arXiv:2507.00029*.

Wenbing Li and 1 others. 2025b. Make LoRA Great Again: Aligning LoRA Mixtures for Efficient Adaptation. *arXiv preprint arXiv:2502.16894*.

Liyuan Liu, Chengyu Dong, Xiaodong Liu, Bin Yu, and Jianfeng Gao. 2023. Bridging Discrete and Backpropagation: Straight-Through and Beyond. *Advances in Neural Information Processing Systems*, 36:12291–12311.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training Language Models to follow Instructions with Human Feedback. *Advances in neural information processing systems*, 35:27730–27744.

Zhen Shao and 1 others. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously Large Neural Networks: The Sparsely-gated Mixture-of-Experts Layer. *arXiv preprint arXiv:1701.06538*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural Network Acceptability Judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Xun Wu, Shaohan Huang, and Furu Wei. 2024. Mixture of LoRA Experts. *arXiv preprint arXiv:2404.13628*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2025. Qwen2.5 Technical Report. *Preprint*, arXiv:2412.15115.

Ziyu Zhao, Tao Shen, Didi Zhu, Zexi Li, Jing Su, Xuwu Wang, Kun Kuang, and Fei Wu. 2024. Merging LoRAs like Playing LEGO: Pushing the Modularity of LoRA to Extremes Through Rank-wise Clustering. *arXiv preprint arXiv:2409.16167*.

Yun Zhu, Nevan Wichers, Chu-Cheng Lin, Xinyi Wang, Tianlong Chen, Lei Shu, Han Lu, Canoee Liu, Liangchen Luo, Jindong Chen, and 1 others. 2023. SiRA: Sparse Mixture of Low-Rank Adaptation. *arXiv preprint arXiv:2311.09179*.

A Detailed Experimental Results

This section provides the **full per-task** results corresponding to the averaged metrics reported in the main paper. All experiments follow the two-stage pipeline: training standalone experts (Stage I) and integrating them through routing (Stage II).

A.1 Stage I: Standalone Expert Quality

Table 3 shows the performance of experts before mixture integration. We compare the original instruction-tuned model (Baseline), conventional fine-tuning (SFT), and RLVF.

The detailed numbers match the observations in Section 5.1. RLVF improves reasoning-heavy tasks for 3B and 8B models, while fine-tuning can reduce performance in larger instruction-tuned models. By contrast, for the 1B model, RLVF gives smaller gains due to limited reasoning capacity.

A.2 Stage II: Expert Integration

Tables 4 and 5 report mixture integration results for both fine-tuned experts and RLVF-trained experts.

We compare LoRAMixer top-1 routing, LoRAMixer normalized top-2 routing, and ours (**Hard-Routed MoR-LoRA**). Across both expert types, hard routing achieves better performance while training far fewer parameters. This indicates that integration mainly depends on selecting the correct expert rather than modifying expert weights.

Summary

Across all tables, three patterns are consistent:

- RLVF produces stronger experts for capable models.
- Soft routing requires retraining experts and more parameters.
- Hard routing preserves expert behavior with fewer trainable parameters.

These tables confirm that the improvements in the main paper are consistent across datasets and model sizes.

B Impact of LoRA Rank in Integration

As described in Section 4.2, Stage II integrates pretrained experts by training a shared linear router and applying LoRA adaptation to the attention layers. The router operates at the token level and selects one expert per token, while in standalone experts, the same adapter processes all tokens of a prompt. This change in routing granularity can alter token representations, so a small attention adaptation is introduced to improve the compatibility between token embeddings and routing decisions.

In this section, we analyze whether Stage II requires substantial parameter adaptation to learn expert selection. We vary the LoRA rank applied to the attention layers. The *No LoRA* setting trains only the shared router while keeping the model representations unchanged.

Table 6 reports the results when integrating the RLVF-trained experts from Stage I. For both model sizes, allowing a small amount of attention adaptation improves performance. However, the gains quickly saturate. For Llama-8B, $R = 32$ gives the highest score (79.93), while the *No LoRA* setting already achieves 78.65, which is close to the maximum.

These results indicate that Stage II primarily learns routing behavior rather than relearning domain knowledge or reasoning ability. The attention LoRA stabilizes token-level routing, but only a small rank is sufficient, and larger ranks provide little additional benefit.

C Results on Additional Model Families

In addition to the LLaMA models used in the main experiments, we also evaluate our method on two other model families: Gemma-3 (Gemma Team et al., 2025) and Qwen-2.5 (Yang et al., 2025). The goal of this experiment is to verify that the observations reported in the main paper are not limited to the LLaMA architecture.

Tables 7 and 8 report the results for Gemma-3-4B and Qwen-2.5-7B, respectively. Table 7 shows the standalone expert performance, while Table 8 reports the integration results. As in the LLaMA experiments, RLVF produces stronger experts, and our method achieves the best integration performance while training far fewer parameters than LoRAMixer.

Overall, these results show that the main findings of the paper remain consistent across different model families. This suggests that the proposed routing framework is general and can be applied to any instruction-tuned architectures, not only the LLaMA family.

D Comparison with Static Approaches

In Stage II, our method integrates the experts dynamically. This means the suitable LoRA expert

Method	GSM8K	ARC-C	MedQA	BoolQ	CoLA	Avg
`Llama-1B`						
Baseline	03.49	22.78	26.71	30.31	28.86	22.43
SFT	**30.10**	**46.07**	**38.17**	61.93	**69.61**	**<u>49.18</u>**
RLVF	00.05	30.12	34.33	**68.23**	30.30	32.61
`Llama-3B`						
Baseline	01.82	22.18	33.70	48.93	60.79	33.48
SFT	60.96	69.97	50.14	83.43	**79.00**	68.60
RLVF	**75.36**	**76.88**	**51.13**	**84.43**	75.65	**<u>72.69</u>**
`Llama-8B`						
Baseline	63.84	81.38	64.65	82.53	76.99	73.89
SFT	70.74	79.52	60.10	73.15	**80.09**	72.72
RLVF	**83.02**	**82.94**	**66.06**	**87.68**	78.91	**<u>79.72</u>**

Table 3: **Performance of standalone fine-tuning (SFT) and reinforcement learning from verifiable feedback (RLVF).** The Baseline row corresponds to the original model. RLVF improves reasoning performance for capable models (3B and 8B) while preserving chain-of-thought behavior, whereas fine-tuning can degrade CoT ability in large instruction-tuned models by overriding existing reasoning patterns. Smaller models (1B) benefit less from RLVF due to limited reasoning trajectory diversity.

Method	GSM8K	ARC-C	MedQA	BoolQ	CoLA	Average	# Trainable Parameters
`Llama-1B`							
LoRAMixer TopK=1	30.93	56.76	25.37	39.72	02.68	31.09	$\approx 315M$
LoRAMixer TopK=2 Normalized	33.89	45.82	30.95	43.55	65.10	43.86	$\approx 315M$
Ours	34.72	45.05	33.31	53.18	68.65	**46.98**	$\approx 27M$
`Llama-3B`							
LoRAMixer TopK=1	22.74	46.50	46.50	75.75	70.57	52.41	$\approx 606M$
LoRAMixer TopK=2 Normalized	63.00	66.13	50.16	80.65	74.78	66.94	$\approx 606M$
Ours	62.85	66.98	49.96	82.00	76.22	**67.60**	$\approx 73M$

Table 4: **Stage II. Integration using fine-tuning experts.** LoRAMixer retrains experts and requires substantially more trainable parameters, while Hard-Routed MoR-LoRA freezes experts and learns only routing and attention adaptation. Despite using fewer parameters, hard routing achieves comparable or better performance, showing that correct expert selection is more important than modifying expert weights.

is selected based on the input during **inference**. In contrast, some methods integrate multiple experts into a single LoRA adapter during the integration stage and then use this merged adapter for all inputs during inference.

One such method is LoRAHub (Huang et al., 2023). LoRAHub learns a set of weights that linearly combine several pretrained LoRA adapters into **one** merged adapter. These weights are optimized using a small dataset to minimize the supervised loss. After this optimization, the merged adapter is fixed and applied to every input during inference.

Table 9 compares our method with LoRAHub. The results show that our dynamic routing approach consistently outperforms the static LoRAHub integration. This suggests that selecting experts based on the input is more effective than merging all experts into a single adapter.

We also observed that LoRAHub required many optimization iterations to achieve good performance (around 300 iterations in our experiments). In contrast, our method trains the router with a single pass over the data during Stage II, resulting in a much faster integration process.

Method	GSM8K	ARC-C	MedQA	BoolQ	CoLA	Average	# Trainable Parameters
`Llama-3B`							
LoRAMixer TopK=1	63.53	68.34	52.40	76.67	67.40	65.67	$\approx 606M$
LoRAMixer TopK=2 Normalized	69.67	73.98	52.08	82.87	75.46	70.81	$\approx 606M$
Ours	73.62	75.43	51.53	83.09	76.70	**72.07**	$\approx 73M$
`Llama-8B`							
LoRAMixer TopK=1	79.51	83.30	65.22	83.70	76.99	77.74	$\approx 1.133B$
LoRAMixer TopK=2 Normalized	82.15	83.96	66.06	82.40	77.08	78.33	$\approx 1.133B$
Ours	84.69	84.61	67.17	84.43	78.09	**79.80**	$\approx 109M$

Table 5: **Stage II. Integration using RLVF-trained reasoning experts.** Hard-Routed MoR-LoRA preserves expert behavior while requiring far fewer trainable parameters, outperforming soft routing methods that retrain experts and modify their learned reasoning patterns.

LoRA Rank	GSM8K	ARC-C	MedQA	BoolQ	CoLA	Average
`Llama-3B`						
No Attention LoRA	63.46	70.82	53.62	81.44	68.19	67.51
R=16	65.81	75.26	51.61	82.17	72.29	69.43
R=32	70.74	76.79	52.63	83.30	75.36	71.76
R=64	72.10	76.79	52.32	83.55	75.55	72.06
R=128	73.62	75.43	51.53	83.09	76.70	**72.07**
`Llama-8B`						
No Attention LoRA	79.76	84.81	65.51	86.09	77.06	78.65
R=16	83.85	85.24	65.52	85.57	78.04	79.64
R=32	83.09	85.67	66.93	86.67	77.28	**79.93**
R=64	85.06	85.67	66.38	84.59	77.28	79.80
R=128	84.69	84.61	67.17	84.43	78.09	79.80

Table 6: **Effect of attention LoRA rank during Stage II integration using RLVF-trained experts.** The configuration of *No LoRA* trains only the shared router while keeping the backbone unchanged. Small ranks improve performance, but gains quickly saturate, indicating that integration mainly depends on routing rather than relearning expert knowledge.

Method	GSM8K	ARC-C	MedQA	BoolQ	CoLA	Avg
`Gemma-3-4B`						
Baseline	85.43	82.62	44.79	80.98	80.63	74.89
SFT	64.97	72.70	42.97	**87.07**	82.84	70.11
RLVF	**88.78**	**84.99**	**47.65**	85.14	**82.07**	**77.73**
`Qwen-2.5-7B`						
Baseline	54.13	89.07	57.42	81.41	83.03	73.01
SFT	77.48	88.14	**57.89**	87.83	**83.99**	79.07
RLVF	**91.66**	**89.48**	57.82	**88.87**	83.13	**82.19**

Table 7: **Performance of standalone experts on additional model families.** We report results for Gemma-3-4B and Qwen-2.5-7B across five tasks. As in the LLaMA experiments, RLVF improves reasoning performance compared to the baseline models.

METHOD	GSM8K	ARC-C	MEDQA	BOOLQ	COLA	AVERAGE	# TRAINABLE PARAMETERS
`Gemma-3-4B`							
LORAMIXER TOPK=1	86.05	83.70	46.50	80.94	80.44	75.53	$\approx 836M$
LORAMIXER TOPK=2 NORMALIZED	86.88	83.10	47.68	83.43	81.30	76.48	$\approx 836M$
OURS	87.57	83.61	48.78	84.80	81.59	**77.27**	$\approx 72M$
`Qwen-2.5-7B`							
LORAMIXER TOPK=1	74.91	88.99	57.50	83.00	83.13	77.51	$\approx 1.211B$
LORAMIXER TOPK=2 NORMALIZED	90.60	88.65	56.20	85.54	81.78	80.55	$\approx 1.211B$
OURS	90.67	89.25	58.13	88.10	83.41	**81.95**	$\approx 41M$

Table 8: **Stage II integration results on additional model families.** We compare LoRAMixer and Hard-Routed MoR-LoRA when integrating pretrained experts. For Qwen-2.5-7B, the mixer is trained with LoRA rank $r = 64$.

METHOD	GSM8K	ARC-C	MEDQA	BOOLQ	COLA	AVERAGE
`Llama-3B`						
LORAHUB	72.55	68.86	51.06	74.20	72.10	67.75
OURS	**73.62**	**75.43**	**51.53**	**83.09**	**76.70**	**72.07**
`Llama-8B`						
LORAHUB	82.41	**84.64**	65.51	76.21	70.04	75.76
OURS	**84.69**	84.61	**67.17**	**84.43**	**78.09**	**79.80**

Table 9: **Stage II comparison with the static integration baseline LoRAHub.** Our dynamic routing approach consistently outperforms LoRAHub static adapter merging across tasks and model sizes.