Tri-LoRAMoE: Mitigating Cross-Domain Interference via Transfer Gain-Aware Knowledge Allocation

Anonymous ACL submission

Abstract

Low-Rank Adaptation(LoRA) efficiently fine-tunes large language models, while LoRA-MoE further extends it to multitask adaptation through sparse expert selection. However, conflicting gradient update directions between heterogeneous tasks cause severe cross-domain interference. Existing LoRA-MoE methods cannot solve this bottleneck, either because routing noise corrupts expert representations due to coupled updates between routes and experts, or because they force task isolation at the expense of expert collaboration. Therefore, we interpret multi-task learning as a transfer gain-aware knowledge allocation process and propose *Tri-LoRAMoE*, a two-stage training framework characterized by a tri-level coordination of transfer gain-aware diagnosis, expert personalization, and metacognitive routing. During the first stage, a general expert gathers shared experience, determining whether individual experts inherit shared prior knowledge by diagnosing net transfer gains. In the second stage, the experts are frozen to train a metacognitive router, an advanced mechanism directing knowledge invocation post-expertise formation. Utilizing task-guided routing and depth-aware penalties, it smoothly transitions the model's behavior from generalized understanding to specialized task reasoning. Experiments on scientific and commonsense reasoning benchmarks demonstrate that Tri-LoRAMoE enhances the average performance of multi-task learning by alleviating cross-domain interference, while preserving unseen-task generalization. [1]

1 Introduction

Pre-trained models exhibit low intrinsic dimensionality, and expanding their solution space typically requires increasing the number of parameters (Li et al., 2018). This observation aligns with the Lottery Ticket Hypothesis (Frankle and Carbin, 2018), which motivates Parameter-Efficient Fine-Tuning (PEFT) methods. PEFT techniques freeze most pre-trained parameters and update only a small set of learnable modules, enabling efficient adaptation and transfer across tasks. Among them, LoRA (Hu et al., 2022) mitigates deployment challenges (Aghajanyan et al., 2021) by updating low-rank weight matrices. While highly effective in single-task settings, LoRA still requires manual adapter selection for multitask learning, limiting its scalability.

Single-agent systems prone to Degeneration-of-Thought (DoT) often fail to revise early incorrect judgments, as shown in (Liang et al., 2024) and (Yin et al., 2023). Mixture-of-Experts (MoE) architectures alleviate this issue via routing mechanisms that coordinate expert collaboration (Fedus et al., 2022). However, conventional MoE models suffer from excessive parameter updates during multitask fine-tuning due to token-level routing and load-balancing constraints. MoLoRA (Zadouri et al., 2023) addresses this by integrating MoE with PEFT, reducing update costs to below 1% while maintaining strong multitask performance.

Although recent methods improve performance, existing LoRA-MoE frameworks still face cross-domain interference. On one hand, most methods that synchronously update experts and routers rely on token IDs with weak contextual relevance to make routing decisions (Xue et al., 2024). Consequently, identical tokens from different task domains often trigger identical routing assignments, leading heterogeneous tasks to compete for the same expert parameters and intensifying task conflict as shown in Figure 1. On the other hand, in two-stage isolated training variants, the first stage is typically designed only for task partitioning. Although this enhances specialization, these approaches largely overlook cross-task knowledge transfer (Huang et al., 2023; Zeng et al., 2023; Feng et al., 2024), which restricts their generalization ca-

[1] Our code is submitted as supplementary material.

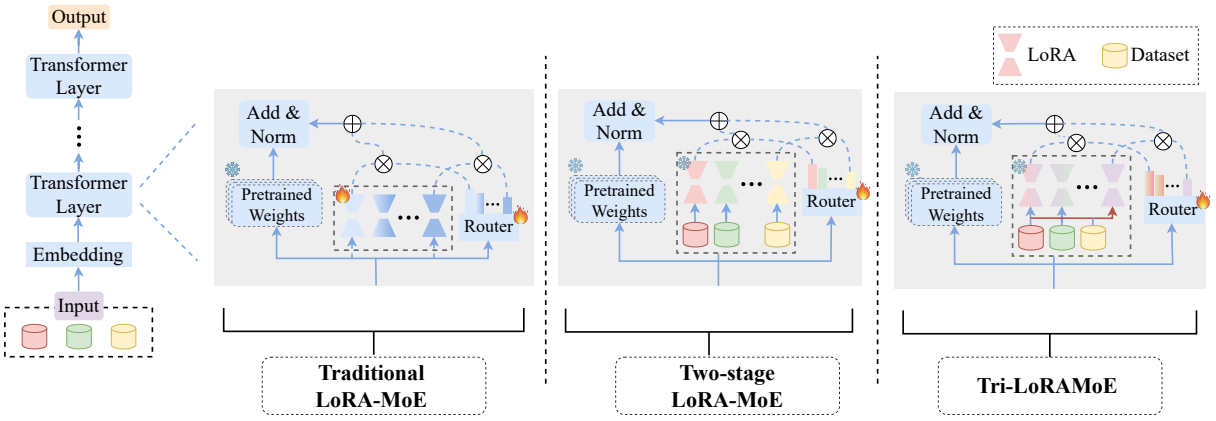


Figure 1: Comparison of three LoRA-MoE training paradigms. Traditional LoRA-MoE jointly updates experts and the router. Standard two-stage LoRA-MoE first trains task-specific experts and then trains the router. In Tri-LoRAMoE, colored cylinders denote task datasets, and colored trapezoids denote LoRA experts. The color gradients inside the experts illustrate knowledge transfer from the general expert to task-specific experts.

pabilities despite their high parameter efficiency.

To address the cross-domain interference problem, we propose Tri-LoRAMoE. It reinterprets LoRA-MoE multi-task learning as a process of knowledge allocation, rather than a pure routing isolation problem. The core question is no longer focusing on how experts are specialized, but what Knowledge is shared, whether this knowledge should be transferred, and how the model should invoke general experts or specialized experts during the inference process.

Specifically, this cognitive process is realized through a two-stage paradigm. The first stage controls the commonality and transferability of knowledge. It first accumulates general experience in the shared expert, diagnoses the net transfer gains between tasks, and then determines whether task-specific experts should inherit prior knowledge or choose independent learning. In Stage 2, freezing all learned experts allows us to optimize the metacognitive router within a stable representation space, thereby enabling high-level routing decisions that are free from expert-knowledge interference. To achieve precise path control, we formulate two complementary architectural constraints. Specifically, task-guided path calibration aligns expert selection with task identity during training to maintain categorical consistency. Meanwhile, depth-aware de-inertia progressively suppresses excessive reliance on the general expert in deeper layers to promote task-specific reasoning. Therefore, Tri-LoRAMoE transforms the model into an adaptive system, smoothly transitioning from generalized understanding to task-specific reasoning.

Our main contributions are summarized as follows:

- We reinterpret multi-task LoRA-MoE as a transfer gain-aware knowledge allocation process, decoupling expert and router training for rational knowledge distribution.
- We introduce a zero-cost transfer criterion for adaptive expert initialization alongside an input-driven metacognitive routing mechanism governed by task-guided and depth-aware constraints.
- Extensive evaluations show that Tri-LoRAMoE outperforms strong baselines across diverse benchmarks while maintaining robust generalization to unseen tasks.

2 Related Work

2.1 Integrating LoRA and MoE

Cross-domain Interference In the LoRA-MoE line of work, MoLoRA (Zadouri et al., 2023) established the first standardized framework by freezing the base model and jointly training the router and expert modules. Despite improvements, LoRA-MoE architectures continue to face challenges, with cross-domain interference being a primary obstacle. LoRAMoE (Dou et al., 2024) mitigates catastrophic forgetting through local balance constraints but ignores task heterogeneity, leaving cross-domain interference unresolved. MoA (Feng et al., 2024) assigns a dedicated expert to each task to reduce conflict, but this design sacrifices inter-task similarity modeling and generalization,

limiting the benefits of multitask collaboration. Likewise, TT-LoRA MoE (Kunwar et al., 2025) adopts MoA's two-stage training with tensor optimizations for memory and speed, yet inherits MoA's limitations without addressing the underlying paradigm constraints.

Although two-stage LoRA-MoE training already separates expert learning from router learning, existing methods such as MoA, TT-LoRA MoE, and related variants still treat Stage 1 mainly as task-wise specialization. Our difference is not merely adding another training trick. We explicitly redesign both stages. In Stage 1, we introduce a general expert as a transferable prior and selectively use it to initialize task experts when mixed-task training shows positive net transfer. In Stage 2, we optimize the router with explicit task alignment and a depth-aware penalty, rather than relying on unguided routing or simple task-to-expert matching. This makes the framework interference-aware in both knowledge transfer and routing.

Knowledge Differences between Transformer Layers Recent analysis reveals MoE's unique efficiency model: a collaborative framework of mid-level activation, late-stage amplification, and foundation refinement, demonstrating layer-wise knowledge variation in large language models (Li et al., 2025). While MixLoRA (Li et al., 2024) reduces parameter redundancy via attention-layer LoRA and MoE in fully connected layers, it neglects cross-layer knowledge differences. MoLA (Gao et al., 2024) compares expert allocation strategies and preliminarily explores inter-layer knowledge differences, whereas Alpha-LoRA (Qing et al., 2024) dynamically allocates experts per layer quality but predetermines allocation pre-fine-tuning, limiting task adaptability. Our approach achieves precise expert-task matching through task-guided routing loss, with layer-wise penalty loss aligning with the efficiency model's balance of shallow-layer generality and deep-layer specialization (Li et al., 2025).

2.2 Shared Expert

The emergence of DeepSeek-R1 (Guo et al., 2025) has heightened attention to shared experts' role in preserving model generalization. DeepSeek-MoE (Dai et al., 2024) introduced an always-activated shared expert post-sparsification, profoundly influencing subsequent research in (Cai et al., 2024), (Gou et al., 2023), and (Zhao et al., 2024). These works primarily employ shared experts for generalization on unknown tasks rather than resolving cross-domain interference. Compared with the generalization ability provided by the shared expert, we explored its ability as a general expert. Departing from direct shared expert usage, we initialize task-specific experts via a general expert to enhance collaboration efficiency, while layer-wise losses penalize general expert selection to balance cross-layer knowledge differences.

3 Methodology

3.1 Preliminary

LoRA In the field of parameter-efficient fine-tuning (PEFT) of large models, LoRA proposes an innovative method to replace the update of the fully connected layer by training only two low-rank matrices (Hu et al., 2022). For the update of each linear projection layer in a large language model, LoRA can restate its update method as follows,

$$W_0 + \Delta W = W_0 + AB, \tag{1}$$

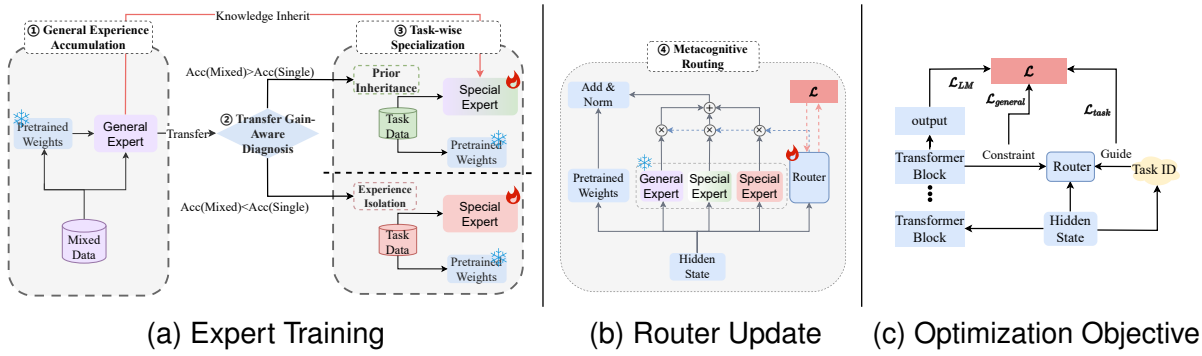
where $W_0 \in R^{d_{out} \times d_{in}}$ represents the parameter matrix of the pre-trained large language model, and $\Delta W \in R^{d_{out} \times d_{in}}$ represents the parameter matrix that needs to be updated if the full fine-tuning method is adopted. $A \in R^{d_{out} \times r}$ and $B \in R^{r \times d_{in}}$ represent the trainable matrices in LoRA.

3.2 Tri-LoRAMoE

Our method mitigates cross-domain interference by decoupling expert formation from expert invocation and by explicitly modeling the transfer decision between them. From a functional viewpoint, Tri-LoRAMoE contains five steps: general experience accumulation, transfer gain-aware diagnosis, prior inheritance or experience isolation, task-wise specialization, and metacognitive routing. Figure 2 groups these operations into expert training, router updates, and optimization objectives.

Expert Training Stage 1 is responsible for building an expert library with both transferable and task-specialized knowledge. As shown in Figure 2a, we first train a general expert over the union of all task data, then decide whether this shared prior should be inherited by each task expert, and finally continue optimizing each task expert on its own dataset.

To accumulate shared experience, all task data are first fed into a single LoRA expert. This expert is not expected to solve every task optimally



(a) Expert Training (b) Router Update (c) Optimization Objective

Figure 2: Specific implementation details of the Tri-LoRAMoE training paradigm

by itself. Instead, it serves as a compact repository of reusable patterns that may later benefit task-specific experts. Let the dataset be $D = \{D_1, D_2, \ldots, D_C\}$, where C denotes the number of tasks. The update rule for the general expert is given by:

$$y = (W_0 + AB)x, x \in D. \quad (2)$$

The next step is transfer gain-aware diagnosis. In practice, whether shared initialization is beneficial depends on the net effect of collaboration and interference across tasks. We therefore use a cost-free criterion based on standard LoRA baselines: if the average accuracy of LoRA(Mixed) exceeds that of LoRA(Single), we treat the task set as transfer-favorable and initialize task-specific experts from the general expert. Otherwise, task-specific experts are trained from independent initialization. This rule operationalizes transfer judgment without task-similarity estimation, auxiliary controllers, or extra optimization.

This diagnostic is intentionally defined at the task-set level rather than the sample level. Our goal in Stage 1 is not to learn a fine-grained transfer map, but to decide whether the multitask mixture as a whole provides a net positive prior for initialization. The comparison between LoRA(Mixed) and LoRA(Single) therefore gives a directly observable estimate of whether collaborative signal outweighs interference, while avoiding another trainable transfer estimator that could itself become unstable or overfit to a small task pool.

After this decision is made, each task expert continues to optimize only on its corresponding dataset. When transfer is beneficial, this step converts a shared prior into a task-specialized adaptation pattern. When transfer is harmful, it preserves independence and prevents negative-transfer contamination. The personalized adaptation process for task-specific updates is as follows:

$$y^i = W_0 x + A^i B^i x, x \in D_i, i = 1, 2, ..., C. \quad (3)$$

Here, A^i and B^i denote the low-rank updates obtained after fine-tuning on the i-th task.

This stage therefore does more than produce isolated task adapters. It constructs a stable expert library in which reusable experience is retained when helpful, isolated when harmful, and finally refined into specialized experts that can be invoked reliably in the second stage.

From an optimization perspective, Stage 1 separates three timescales that are usually conflated in LoRA-MoE training: acquiring reusable knowledge, deciding whether it should be shared, and specializing it for each task. This separation matters because negative transfer is handled before router learning begins. As a result, the second stage receives experts whose differences already reflect transfer gain-aware adaptation rather than artifacts introduced by early routing mistakes.

Router Updates As illustrated in Figure 2b, Stage 2 inserts the acquired experts into each layer alongside randomly initialized routers. By freezing all expert weights and optimizing exclusively the routers, the model effectively learns how to deploy the established expert library rather than continuously rewriting the expert representations themselves. This decoupling successfully prevents routing noise from distorting expert specialization, thereby converting the conventional routing process into an explicit knowledge-control mechanism.

We train the router on uniformly sampled balanced data so that task frequency does not dominate routing behavior. The router therefore learns from a

controlled view of task diversity and can better estimate whether an input should remain on the general path or be dispatched to a specialized expert. Formally, after freezing the general and task-specific experts across the L-layer Transformer, the router at layer l with trainable $W_l \in \mathcal{R}^{d_{in} \times (C+1)}$ selects experts for input x using the top-k method (Shazeer et al., 2017):

$$S_l^k(x) = \frac{TopK(softmax(W_l x), K)_k}{\sum_{k=1}^{K} TopK(softmax(W_l x), K)_k}. \quad (4)$$

Here, $s_l^k(x)$ denotes the normalized probability assigned to the k-th selected expert. In this sense, the router plays the role of a metacognitive scheduler.

Freezing experts also changes what the router must learn. Because expert parameters no longer drift during Stage 2, the router is trained against a stable expert library with fixed capabilities. The resulting routing boundary is therefore easier to interpret: selection probabilities reflect the compatibility between an input and existing experts, rather than a moving target produced by simultaneous expert and router updates.

Optimization Objective Decoupling the two stages can eliminate update interference, but this itself cannot guarantee correct routing. Therefore, as shown in Figure 2c, the optimization objective equips the router with two additional forms of control, which are supervised path calibration and depth-wise de-inertia. The former tells the router which expert family should be preferred during the training process, while the latter prevents deep layers from falling into overly generic solutions. The final loss function consists of the language modeling loss $\mathcal{L}_{LM}$ and the routing loss $\mathcal{L}_{rt}$:

$$\mathcal{L} = \mathcal{L}_{LM}(x) + \alpha \mathcal{L}_{rt}. \quad (5)$$

Here, α controls the contribution of the routing loss $\mathcal{L}_{rt}$, and $\mathcal{L}_{LM}$ is the standard language modeling objective defined as:

$$\mathcal{L}_{LM} = -\sum_{i=1}^{N} logP(x_i|x_{<i}). \quad (6)$$

We plot the normalized average variance of router output weights at each layer when only $\mathcal{L}_{LM}$ is used, as shown in Figure 3. As depth increases, routing becomes increasingly concentrated. However, in the representative final layer, when the general expert is selected as the top-1 expert, its average normalized weight is about 40%, whereas

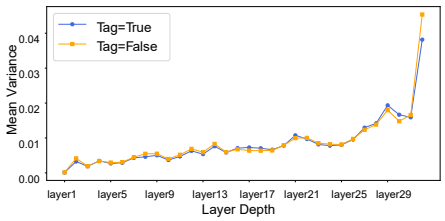


Figure 3: Layer-wise variance distribution of router outputs when only the language modeling loss is used. "Tag=True" and "Tag=False" denote samples whose final predictions are correct or incorrect, respectively. A larger variance indicates more concentrated routing decisions.

the probability of selecting the task-corresponding expert is only 0.07%. This shows that an unguided router easily collapses onto the general expert, especially in deeper layers where specialized reasoning should instead become more important.

To address this behavior, we introduce a task-guided loss and a layer-wise penalty loss. Together they calibrate routing paths and encourage the router to diversify its decisions instead of repeatedly defaulting to the general expert. The router loss consists of two components:

$$\mathcal{L}_{rt} = \mathcal{L}_{task} + \beta \mathcal{L}_{general}, \quad (7)$$

where β controls the update weight ratio between task-guided loss $\mathcal{L}_{task}$ and layer-wise penalty loss $\mathcal{L}_{general}$.

The task-guided loss serves as supervised path calibration. We compute the cross-entropy between router logits and the task identity so that, during training, the router learns which specialized expert should dominate for each task family. Assume that the number of input tasks is C. For an input token with task label y_i, the task-guided loss $\mathcal{L}_{task}$ is defined as follows:

$$\mathcal{L}_{task} = -\frac{1}{N} \sum_{i=1}^{N} log(\frac{exp(z_{i,y_i})}{\sum_{j=1}^{C+1} exp(z_{i,j})}), \quad (8)$$

where $z_{i,j}$ represents the logits output of the j-th expert model selection of the router of the i-th token, $C+1$ represents the number of C tasks and a general expert model, and N represents the total number of tokens.

Notably, task labels are used only as lightweight supervision for router training in the second stage. At inference time, routing is determined entirely by the input representations. For corpora without explicit task labels, pseudo task labels can be

obtained by unsupervised clustering over sample embeddings and then used within the same training pipeline.

The layer-wise penalty loss can be viewed as depth-wise de-inertia. Hinted by Figure 3, instead of suppressing the general expert uniformly, we can progressively increase the penalty across layers so that shallow layers remain free to exploit broad transferable knowledge, whereas deeper layers are increasingly encouraged to invoke task-specific experts. This loss is formulated as:

$$\mathcal{L}_{general} = -\frac{1}{L}\sum_{l=1}^{L} w_l \frac{1}{N}\sum_{i=1}^{N} log(\frac{exp(z_{i,C+1})}{\sum_{j=1}^{C+1} exp(z_{i,j})}), \quad (9)$$

where $w_l = f(l; a, b, L)$ schedules the penalty strength assigned to layer l. According to the intuition of higher penalties for deeper layers, a preliminary implementation can be a linearly increasing function:

$$w_l = a + \frac{b-a}{L-1} \times (l-1). \quad (10)$$

While this simple linear penalty schedule guides a gradual transition from general understanding to specialized reasoning, it may not shape the optimal penalties, which require model- and task-specific tuning.

The resulting training dynamics differ from jointly optimized LoRA-MoE systems: experts stabilize before routing becomes task-sensitive, so early routing noise no longer corrupts expert formation. This makes supervised path calibration more meaningful and helps the model preserve collaboration while avoiding destructive interference.

Load balancing critically impacts sparse MoE performance. Unlike prior LoRA-MoE methods that rely on explicit balancing regularizers, our framework reduces expert dominance through three structural choices: decoupled expert and router training, routing-loss-guided path calibration, and uniform data sampling in Stage 2. These mechanisms stabilize routing behavior without requiring an additional balancing objective.

4 Experiments

4.1 Experimental Setup

Baselines We compare Tri-LoRAMoE with several representative PEFT and MoE-based methods, including LoRA (Hu et al., 2022), LoRAMoE (Dou et al., 2024), MixLoRA (Li et al., 2024), MoLA (Gao et al., 2024), and TT-LoRA MoE (Kunwar et al., 2025). LoRAMoE provides the earliest exploration of cross-domain interference. MixLoRA and MoLA focus on reducing parameter redundancy, and MoLA further investigates layer-wise knowledge distinctions. TT-LoRA MoE integrates tensor parallelism to reduce memory usage. For fairness, we drop its parallelism component and retain only its two-stage training procedure, which is most relevant to our method.

Base Model	Method	Avg.	Time
LLaMa2-7b	LoRA(Single)	76.40	397
	LoRA(Mixed)	77.21	546
	MixLoRA	75.94	1127
	LoRAMoE	77.92	1294
	MoLA	76.68	1894
	TT-LoRA MoE	76.95	1378
	Ours (w/ init)	**80.11**	1065
	Ours (w/o init)	79.44	1065
LLaMa3-8b	LoRA(Single)	83.44	365
	LoRA(Mixed)	79.28	500
	MixLoRA	78.54	1130
	LoRAMoE	82.52	1320
	MoLA	79.22	1777
	TT-LoRA MoE	83.33	1270
	Ours (w/ init)	82.78	932
	Ours (w/o init)	**85.15**	932
LLaMa2-13b	LoRA(Single)	81.22	592
	LoRA(Mixed)	79.99	1182
	MixLoRA	81.45	1509
	LoRAMoE	82.25	2166
	MoLA	81.92	2029
	TT-LoRA MoE	81.46	2100
	Ours (w/ init)	82.27	1885
	Ours (w/o init)	**82.85**	1885

Table 1: Summary of the main results on handling cross-domain interference. All values except Time are reported as percentages, and Time is measured in minutes. Ours (w/o init) denotes training without initializing task-specific experts from the general expert, and Ours (w/ init) denotes training with such initialization. Bold entries indicate the best results, and underlined entries indicate the second best.

Datasets We evaluate all methods on eight benchmarks, including ARC (ARC-c and ARC-e) (Clark et al., 2018), BoolQ (Clark et al., 2019), OBQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaS (Zellers et al., 2019), and WinoG (Sakaguchi et al., 2021). These datasets cover scientific reasoning, multi-step reasoning, social reasoning, physical common sense, factual yes or no questions, and general common-sense inference. For router optimization in stage two, we uniformly sample one thousand instances from each dataset. The dataset is in English and does not involve any specific sensitive

Base Model	Method	Avg.
LLaMa2-7B	LoRA(Mixed)	77.2
	Ours(W/ init)	80.1
	Ours(Unseen)	77.5
LLaMa3-8B	LoRA(Mixed)	79.3
	Ours(W/ init)	82.8
	Ours(Unseen)	80.5
LLaMa2-13B	LoRA(Mixed)	80.0
	Ours(W/ init)	82.3
	Ours(Unseen)	81.0

Table 2: Accuracy on Unseen Tasks and Improvements over LoRA(Mixed)

demographic groups.

Model and Hyperparameter Setup We conduct experiments on LLaMa2-7B, LLaMa2-13B, and LLaMa3-8B. We trained for two rounds, totaling approximately 232k steps. To ensure consistent PEFT scaling, we set the LoRA rank to $r = 16$ with a scaling factor of 32, and insert adapters into both attention and FFN layers except for MixLoRA. All models use a peak learning rate of 2×10^{-4}, except MixLoRA and MoLA on larger LLaMa variants, which use 1×10^{-5}. Loss weights are set to $\alpha = 1$ and $\beta = 0.1$, with layer-wise penalty boundaries $a = 0$ and $b = 0.5$.

Metrics Following standard LLM evaluation practices, we use accuracy (Acc) as the primary metric for all scientific reasoning and common sense reasoning benchmarks. To comprehensively evaluate multi-task capabilities, we report the accuracy for each individual task within the training set and the average accuracy across all tasks in the main experiment. All experiments are conducted on a single NVIDIA A800 GPU

4.2 Main Result

Cross-domain Interference Table 1 summarizes the average performance and efficiency across different backbone scales on commonsense reasoning tasks, while Appendix A reports the dataset-specific scores. LoRA (Single) fine-tunes a dedicated module for each task individually, whereas LoRA (Mixed) jointly trains a single module to handle all tasks. Although neither is a LoRA-MoE approach, their average performance clearly reveals the severity of cross-domain interference. On LLaMa2-7b, LoRA (Mixed) outperforms LoRA (Single) by 0.81 percentage points on average. However, this advantage vanishes on larger backbones, dropping by 4.16 percentage points on LLaMa3-8b and 1.23 percentage points on LLaMa2-13b. These results demonstrate that even with scaled-up training data, LoRA (Mixed) remains heavily constrained by unresolved task interference.

Our method achieves an optimal balance between overall accuracy and efficiency. We introduce two variants of our optimized two-stage LoRA-MoE framework, distinguished primarily by the role of the general expert in Stage 1. The activation of the general expert initialization depends on the relative average accuracy between LoRA (Single) and LoRA (Mixed). Specifically, when $\text{accuracy}_{\text{MIXED}} > \text{accuracy}_{\text{SINGLE}}$, initializing with the general expert proves beneficial, whereas it becomes counterproductive otherwise. This adaptive configuration enables the model to effectively balance knowledge generalization and specialization. Overall, our method (with init) performs best on LLaMa2-7b, while our method (w/o init) achieves the highest scores on LLaMa3-8b and LLaMa2-13b. In terms of average accuracy, our best variant outperforms the strongest baseline by 2.19, 1.71, and 0.60 percentage points on LLaMa2-7b, LLaMa3-8b, and LLaMa2-13b, respectively. Beyond accuracy, our approach also demonstrates superior efficiency compared to most existing LoRA-MoE methods, with the sole exception of LLaMa2-13b where MixLoRA is slightly faster but yields significantly lower average accuracy.

Mitigating cross-domain conflict hinges on resolving the multi-task seesaw phenomenon (Tang et al., 2020), where optimizing certain tasks comes at the expense of others. As shown in Table 5 of Appendix A, taking the LLaMa3-8b backbone as an example, the performance of LoRA (Mixed) on the ARC-e and WinoG datasets drops significantly compared to its single-task counterpart. In contrast to existing LoRA-MoE baselines, our method successfully bypasses this trade-off and consistently secures the highest or second-highest accuracy across nearly all benchmarks.

Cross-task Generalization To evaluate the ability of our method to generalize to unseen tasks, we adopt a leave-one-out setting, using seven datasets as known tasks and treating the remaining one as an unseen task for prediction. For the sake of experimental consistency and simplicity, the general experts of all algorithms in Table 2 (including LoRA(Mixed)) are trained on all datasets. In contrast, in Ours (Unseen), the data of unseen tasks do

w / General expert	Init	Avg.
-	-	77.61%
✓	-	79.44%
✓	✓	80.11%

Table 3: Effect of the general expert and selective initialization on LLaMA2-7B. "General expert" indicates whether a shared expert is trained in Stage 1. "Init" indicates whether task-specific experts are initialized from that general expert.

Layer-wise Penalty Schedule	Avg.
Linear decline	79.87%
Linear rise	80.10%
Quadratic rise	79.59%
Cubic rise	79.79%

Table 4: Average accuracy across varying penalty strength schedules.

not have corresponding task-specific experts and are excluded from router updates. Table 2 summarizes the average results, while Appendix B reports the per-dataset scores without repeating the overall averages and focuses on the held-out datasets that best illustrate either strong transfer or the remaining failure modes. Although the unseen-task setting remains below training on all tasks, Ours (Unseen) still improves over LoRA (Mixed) by 0.3, 1.2, and 1.0 percentage points on LLaMa2-7b, LLaMa3-8b, and LLaMa2-13b, respectively. These results demonstrate the strong overall generalization ability of our approach in handling previously unseen tasks, and Appendix B shows which held-out datasets contribute to the remaining gaps.

4.3 Ablation Study

General Expert To validate the effectiveness of soft-starting specific experts through general expert initialization, we compare the performance of different general expert initialization strategies, whose results are shown in Table 3. To ensure a fair comparison, we strictly followed the two-stage training paradigm, isolating the training of experts from the training of the router. It can be seen that when no general expert is used, the average performance is even lower than that of the LoRAMoE algorithm (see Table 1). However, when the general expert is introduced without using it to initialize the specific expert (General Expert (No Init.)), we observe a significant improvement compared to the no general expert scenario, with an average increase of approximately 1.83 percentage points. Furthermore, when the general expert is used to initialize the specific experts (General Expert (Task-Aware Init.)), the model performance is further enhanced, with an average increase of approximately 2.5 percentage points. This demonstrates that training a general expert already yields better performance than the variant without a general expert, while selective initialization further improves performance. This supports our central claim that the first stage should balance transfer and personalization, rather than enforcing pure task separation.

Layer-wise Penalty Strength Schedule Table 4 presents the effects of different schedules of the layer-wise penalty strength w_l on model performance. We set the penalty threshold between 0 and 0.5. The quadratic and cubic schemes increase the penalty according to quadratic and cubic functions over the layers within the threshold range. In this preliminary exploration, the linearly increasing schedule performs best among the tested options, outperforming quadratic rise and cubic rise. The advantage of linear rise over linear decline therefore serves only as an initial indication that progressively strengthening the penalty across layers may better match the observations in Figure 3, rather than establishing linear growth as a fixed requirement. However, the optimal schedule still requires further exploration in the future. Other ablation experiments can be found in Appendix C.

5 Conclusion and Future Work

Tri-LoRAMoE redefines two-stage LoRA-MoE as a transfer gain-aware knowledge allocation process. In the first stage, a transfer gain-diagnosis determines whether to inherit general priors, followed by knowledge specialization. In the second stage, the router is trained as a metacognitive controller through task guidance and depth inertia mitigation. Experimental results demonstrate that this mechanism effectively suppresses cross-domain interference while promoting task collaboration, significantly enhancing the generalization performance to unseen tasks(other emergent capabilities are further discussed in Appendix D).

More broadly, our results suggest that knowledge sharing in multitask LoRA-MoE is a conditional decision rather than a default assumption. Future work will focus on adaptive routing control, including learned layer-wise penalties, transfer diagnostics, and label-light variants.

Limitations

Our evaluation is limited to three LLaMA family backbone models and eight scientific and commonsense reasoning benchmarks. While these settings are sufficient to test heterogeneous multitasking adaptability under controlled conditions, they do not cover other model families, instruction execution scenarios, multilingual environments, long text generation, safety-critical applications, or broader task distributions. Therefore, our conclusions should be interpreted as evidence of the robustness of LoRA-MoE multitasking adaptability in reasoning-oriented benchmarks, rather than a general guarantee for all LLM adaptability scenarios.

Tri-LoRAMoE also relies on some assumptions that may not always hold true in practice. First, the second-stage training uses task labels as lightweight supervision for route optimization. During the inference stage, routes are determined by input representations rather than task labels, but the training process still assumes the existence of reliable task identifiers. For corpora without explicit task labels, pseudo-labels can be obtained through unsupervised clustering, but this approach is not empirically evaluated in this paper. If task labels or pseudo-labels are noisy, the task-guided routing loss may provide incorrect path supervision, thus undermining expert selection capabilities. At this point, it might be necessary to reduce the weight of the task-guided loss to force the router to rely on input-driven processes.

Secondly, our proposed depth-aware penalty assumption is that shallow layers should retain more general, transferable knowledge, while deeper layers should increasingly rely on task-specific experts. This design is motivated by the observed phenomenon that unguided routes tend to converge towards general expert paths in deeper layers, but the current implementation employs a simple, hand-designed penalty strategy. While preliminary ablation experiments show that the linearly increasing penalty strategy works well in our scenario, the optimal strategy may depend on the backbone network, the degree of task mixing, the number of experts, and route sparsity. Therefore, the current penalty design should be considered an effective choice, but not universally optimal.

Finally, our experiments prioritize comparing accuracy and training time under a fixed scenario rather than conducting a comprehensive deployment-oriented efficiency analysis. We report the training time and compare it with several PEFT and LoRA-MoE baselines, but we do not systematically evaluate inference latency, memory footprint, route sparsity, expert parallelism, or scalability across a wider range of expert numbers. These factors may impact practical deployments, especially in resource-constrained environments. Future work will investigate adaptive migration diagnostics, light-label router supervision, learning-based layer-by-layer penalty scheduling, and more comprehensive deployment costs.

References

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7319–7328.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Weilin Cai, Juyong Jiang, Le Qin, Junwei Cui, Sunghun Kim, and Jiayi Huang. 2024. Shortcut-connected expert parallelism for accelerating mixture-of-experts. *arXiv preprint arXiv:2404.05019*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of NAACL-HLT*, pages 2924–2936.

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*.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, and 1 others. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, and 1 others. 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 (Volume 1: Long Papers)*, pages 1932–1945.

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.

Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Yu Han, and Hao Wang. 2024. Mixture-of-loras: An efficient multitask tuning for large language models. *arXiv preprint arXiv:2403.03432*.

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Baochen Sun, Ruibo Liu, Daiyi Peng, Yawen Zhang, Xiaoyuan Guo, Jie Yang, and VS Subrahmanian. 2024. Higher layers need more lora experts. *arXiv preprint arXiv:2402.08562*.

Yunhao Gou, Zhili Liu, Kai Chen, Lanqing Hong, Hang Xu, Aoxue Li, Dit-Yan Yeung, James T Kwok, and Yu Zhang. 2023. Mixture of cluster-conditional lora experts for vision-language instruction tuning. *arXiv preprint arXiv:2312.12379*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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*.

Pradip Kunwar, Minh N Vu, Maanak Gupta, Mahmoud Abdelsalam, and Manish Bhattarai. 2025. Tt-lora moe: Unifying parameter-efficient fine-tuning and sparse mixture-of-experts. *arXiv preprint arXiv:2504.21190*.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. 2018. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*.

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*.

Junzhuo Li, Bo Wang, Xiuze Zhou, Peijie Jiang, Jia Liu, and Xuming Hu. 2025. Decoding knowledge attribution in mixture-of-experts: A framework of basic-refinement collaboration and efficiency analysis. *arXiv preprint arXiv:2505.24593*.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2024. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17889–17904.

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*, pages 2381–2391.

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*, pages 20511–20523.

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

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. In *Conference on Empirical Methods in Natural Language Processing*.

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*.

Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. 2020. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *Proceedings of the 14th ACM conference on recommender systems*, pages 269–278.

Fuzhao Xue, Zian Zheng, Yao Fu, Jinjie Ni, Zangwei Zheng, Wangchunshu Zhou, and Yang You. 2024. Openmoe: An early effort on open mixture-of-experts language models. In *International Conference on Machine Learning*, pages 55625–55655. PMLR.

Zhangyue Yin, Qiushi Sun, Cheng Chang, Qipeng Guo, Junqi Dai, Xuan-Jing Huang, and Xipeng Qiu. 2023. Exchange-of-thought: Enhancing large language model capabilities through cross-model communication. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15135–15153.

Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermiş, Acyr Locatelli, and Sara Hooker. 2023. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. *arXiv preprint arXiv:2309.05444*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

Weihao Zeng, Lulu Zhao, Keqing He, Ruotong Geng, Jingang Wang, Wei Wu, and Weiran Xu. 2023. Seen to unseen: Exploring compositional generalization of multi-attribute controllable dialogue generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14179–14196.

Hao Zhao, Zihan Qiu, Huijia Wu, Zili Wang, Zhaofeng He, and Jie Fu. 2024. Hypermoe: Towards better mixture of experts via transferring among experts. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10605–10618.

A Per-Dataset Analysis of Main Results

Table 5 reports the per-dataset scores corresponding to the summary in Table 1. The overall averages remain in the main text. Rather than discussing every benchmark exhaustively, we focus on the representative gains and informative exceptions that best explain why Tri-LoRAMoE improves the overall averages.

OBQA OBQA is the clearest case where our method dominates across scales. The best Tri-LoRAMoE variant leads all baselines on every backbone, and the preferred variant shifts from w/ init on LLaMa2-7b to w/o init on the larger backbones. This shows that our framework consistently improves multi-step reasoning under heterogeneous task mixtures.

WinoG and PIQA WinoG and PIQA illustrate strong positive transfer. On WinoG, our method is best on all three backbones, with especially large gains on LLaMa2-7b and LLaMa3-8b, indicating that the routing strategy helps with referential disambiguation. PIQA shows the same cross-backbone pattern: the best Tri-LoRAMoE variant beats the strongest prior LoRA-MoE baseline in every setting, supporting the claim that the framework transfers well to physical commonsense reasoning.

ARC-c and BoolQ These datasets expose the main boundaries of the method. ARC-c remains challenging: our method is strongest on LLaMa2-7b and LLaMa3-8b, but prior LoRA-MoE baselines are still better on LLaMa2-13b. BoolQ is the main exception to otherwise broad gains, since LoRA(Single) is still slightly higher than Ours (w/o init) on LLaMa3-8b. Together, these cases clarify that Tri-LoRAMoE is most effective when transfer opportunities outweigh task-format mismatch or unusually difficult reasoning requirements.

Initialization Pattern Across datasets, w/ init is more effective on LLaMa2-7b, whereas w/o init dominates on LLaMa3-8b and LLaMa2-13b. This mirrors the average-level criterion discussed in the main text and shows that the usefulness of shared initialization depends on backbone scale and task heterogeneity.

B Per-Dataset Analysis of Unseen Tasks

Table 6 reports the per-dataset leave-one-out results corresponding to the summary in Table 2. The overall averages remain in the main text. We focus on the held-out datasets that most clearly demonstrate either strong positive transfer or the remaining failure modes.

ARC-e and ARC-c These two science benchmarks provide the clearest evidence of generalization. Ours (Unseen) improves over LoRA(Mixed) on all three backbones for both ARC-e and ARC-c, with the largest jump appearing on ARC-c for LLaMa3-8b. This suggests that the shared expert and router retain transferable reasoning structure even when a dedicated task-specific expert is absent.

PIQA PIQA also generalizes consistently well. Ours (Unseen) surpasses LoRA(Mixed) on all three backbones, showing that physical commonsense reasoning is effectively supported by the shared expert at test time and helping explain why the average unseen-task gains remain positive across scales.

OBQA, WinoG, and BoolQ These datasets reveal the remaining boundary cases. OBQA is slightly worse on LLaMa2-7b and tied on LLaMa3-8b before turning positive on LLaMa2-13b, so multi-step reasoning transfer appears more sensitive to model scale. WinoG is similarly mixed, with slight drops on LLaMa2-7b and LLaMa2-13b but a large gain on LLaMa3-8b. BoolQ underperforms on LLaMa2-7b yet improves on the larger backbones, which is consistent with the binary classification mismatch already observed in the main results.

Overall Pattern The unseen-task drops are sparse and bounded, and they are concentrated mainly on LLaMa2-7b. Positive transfer dominates on the larger backbones, which explains why the average gains in Table 2 increase with model scale.

C Additional Ablation Study

Sensitivity Analysis To evaluate the robustness of our algorithm, we conduct a sensitivity analysis on the weights of different components in the loss function. We first examine the model's sensitivity to the layer-wise penalty loss term (i.e., $\mathcal{L}_{\text{general}}$). As shown in Figure 4a, we vary the weight β in the set 0, 0.01, 0.1, 1, 10. It can be observed that the model achieves the highest accuracy of 80.11% when β is set to 1, while the lowest accuracy is observed when the weight is 0. This reveals that, in

Base Model	Method	ARC-e	OBQA	SIQA	ARC-c	WinoG	PIQA	BoolQ	HellaS
LLaMa2-7b	LoRA(Single)	77.44	79.60	78.35	52.90	75.85	80.96	73.61	92.46
	LoRA(Mixed)	79.55	80.40	78.45	65.53	73.72	79.54	69.11	91.37
	MixLoRA	77.82	78.60	76.97	64.51	73.64	80.14	65.08	90.76
	LoRAMoE	80.47	80.60	79.07	65.19	73.32	81.77	70.58	92.36
	MoLA	78.54	77.80	78.76	65.53	73.95	81.45	66.88	90.50
	TT-LoRA MoE	76.43	79.60	78.56	58.28	75.69	81.01	73.36	92.66
	Ours (w/ init)	**82.70**	81.20	79.94	**68.00**	**80.00**	**83.51**	72.57	92.96
	Ours(w/o init)	79.55	**82.80**	**80.81**	64.93	76.95	82.81	**73.85**	**93.85**
LLaMa3-8b	LoRA(Single)	86.32	84.40	79.89	76.37	84.21	87.05	**75.26**	94.00
	LoRA(Mixed)	83.00	80.20	77.99	69.62	76.16	83.51	70.31	93.48
	MixLoRA	80.81	77.80	77.99	69.11	77.27	83.84	69.11	92.41
	LoRAMoE	86.49	85.20	78.51	76.54	80.66	87.05	72.05	93.68
	MoLA	80.72	78.80	78.71	69.62	80.51	85.85	67.71	91.82
	TT-LoRA MoE	86.28	84.20	79.68	77.05	83.90	87.21	74.31	93.97
	Ours (w/ init)	85.44	85.60	79.68	73.21	83.50	86.07	74.07	94.68
	Ours (w/o init)	**88.34**	**86.60**	**81.42**	**80.03**	**85.32**	**88.47**	75.05	**95.97**
LLaMa2-13b	LoRA(Single)	82.95	82.80	80.09	67.49	82.32	85.09	74.43	94.57
	LoRA(Mixed)	82.58	80.20	79.73	70.05	78.45	83.13	71.65	94.13
	MixLoRA	84.18	84.20	79.79	72.10	77.82	84.33	74.04	95.12
	LoRAMoE	84.81	85.00	**81.68**	**73.98**	80.19	84.17	73.76	94.42
	MoLA	83.71	84.20	**81.68**	73.38	79.56	84.11	73.67	95.03
	TT-LoRA MoE	82.41	83.80	80.14	69.11	82.08	85.20	74.34	94.59
	Ours (w/ init)	**85.06**	82.80	80.40	71.93	82.72	85.75	74.71	94.76
	Ours (w/o init)	84.01	**85.80**	81.47	71.25	**83.43**	**85.85**	**75.23**	**95.78**

Table 5: Detailed experimental results for each subset of datasets.

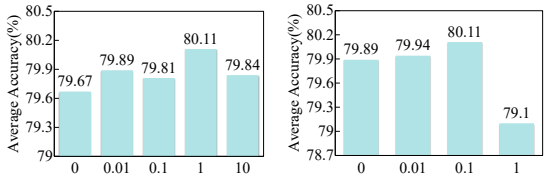
Base Model	Method	ARC-e	OBQA	SIQA	ARC-c	WinoG	PIQA	BoolQ	HellaS
L2-7b	LoRA(Mixed)	79.6	80.4	78.5	65.5	73.7	79.5	69.1	91.4
	Ours(W/ init)	82.7	81.2	79.9	68.0	80.0	83.5	72.6	93.0
	Ours(Unseen)	80.8(1.2↑)	80.0(0.4↓)	80.0(1.5↑)	66.5(1.0↑)	72.2(0.5↓)	80.1(0.6↑)	68.0(1.1↓)	92.1(0.7↑)
L3-8b	LoRA(Mixed)	83.0	80.2	78.0	69.6	76.2	83.5	70.3	93.5
	Ours(W/ init)	85.4	85.6	79.7	73.2	83.5	86.1	74.1	94.7
	Ours(Unseen)	84.3(1.3↑)	80.2(0.0)	78.8(0.8↑)	72.4(2.8↑)	78.3(2.1↑)	84.9(1.4↑)	70.9(0.6↑)	94.1(0.6↑)
L2-13b	LoRA(Mixed)	82.6	80.2	79.7	70.1	78.5	83.1	71.7	94.1
	Ours(W/ init)	85.1	82.8	80.4	71.9	82.7	85.8	74.7	94.8
	Ours(Unseen)	85.2(2.6↑)	80.8(0.6↑)	79.8(0.1↑)	71.3(1.2↑)	78.4(0.1↓)	84.8(1.7↑)	73.3(1.6↑)	94.5(0.4↑)

Table 6: Detailed results for each task were not found on each subset of the dataset.

the absence of the general loss term, the selection of the general expert model is unconstrained, leading to a tendency for the model to over-rely on deep experts, exhibiting a form of selection laziness.

For the parameter α (Figure 4b), performance remains stable within the range $0 \leq \alpha \leq 0.1$, and achieves the best result at $\alpha = 0.1$. When $\alpha = 1$, performance drops to 79.1%. This degradation reflects over-regularization rather than an inherent limitation of the method, indicating that excessively strong constraints on router expert selection can impair the model's predictive capability.

Sample Size In this subsection, we investigate the impact of varying training sample sizes on the test accuracy using LLaMa2-7B as the base model. For datasets whose total number of training samples is smaller than the specified sampling amount, we use all available data for training (note: some datasets contain fewer than 2,000 training samples). As shown in Figure 5, although the training time



(a) Sensitivity of β. (b) Sensitivity of α.

Figure 4: The sensitivity of router and layer-wise penalty loss weights α and β.

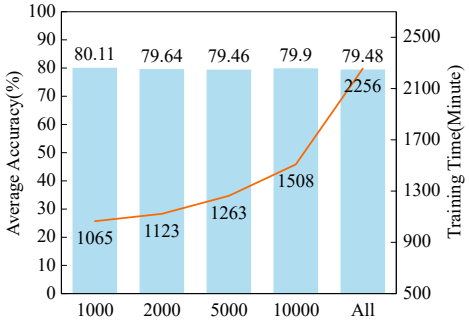


Figure 5: Test accuracy and training time across varying training sample sizes.

increases with the number of samples, the test accuracy does not improve accordingly. This result indicates, on the one hand, that training the router does not require a large volume of data but rather sufficient task-related information. On the other hand, using 1,000 samples achieves a well-balanced expert selection. Injecting more samples may introduce data imbalance, causing the router to overfit. This effect is most evident when the entire dataset is used.

D Discussion

Scalability Compared with existing LoRA-MoE algorithms, our two-stage training paradigm offers greater flexibility and lower coupling. Traditional LoRA-MoE methods typically require joint training of the router and all expert modules when new task data arrive. When the new and old datasets differ substantially, these methods often need to retrain the entire model on the combined data to avoid old-expert bias, where previously trained experts fail to adapt to the new task. In contrast, our framework adapts to new tasks more efficiently. When task conflicts are severe and general knowledge transfer is not applied, only the corresponding task-specific expert and the lightweight second-stage router need to be retrained. When task conflicts are mild, it suffices to incrementally update the general expert: the general expert initializes the task-specific expert, which is then trained accordingly, while the router undergoes incremental refinement. Retraining all task-specific experts is optional rather than required.

Initialization Sensitivity The effectiveness of general expert initialization varies with model size, task complexity, and cross-task interference. For smaller backbones such as LLaMA2-7B and tasks of moderate complexity with partial correlations, a shared initialization creates a common representation space that enables effective knowledge transfer and improves generalization. However, for larger models like LLaMA3-8B or LLaMA2-13B, or for tasks with more diverse reasoning patterns, styles, or domains, a unified initialization becomes restrictive. In these settings, task-specific experts need independent representations to capture fine-grained information, and forcing a shared initialization may lead to mild negative transfer.

Transferability and Integration Cost Although our experiments focus on the LLaMA series, the proposed framework is inherently architecture agnostic and can be seamlessly applied to other large language models. Mainstream transformer architectures, such as DeepSeek, Qwen2.5, and LLaMA, share a common structural pattern based on linear projection submodules (such as Q, K, V). Our LoRA and MoE components operate exclusively on these projection layers and do not modify other architectural elements such as attention heads, normalization schemes (RMSNorm or LayerNorm), or residual pathways.