Mixture of Routers

Anonymous ACL submission

Abstract

Supervised fine-tuning (SFT) is a milestone in aligning large language models with human instructions and adapting them to downstream tasks. In particular, Low-Rank Adaptation (LoRA) has gained widespread attention due to its parameter efficiency. However, its impact on improving the performance of large models remains limited. Recent studies suggest that combining LoRA with Mixture-of-Experts (MoE) can significantly enhance fine-tuning performance. MoE adapts to the diversity and complexity of datasets by dynamically selecting the most suitable experts, thereby improving task accuracy and efficiency. Despite impressive results, recent studies reveal issues in the MoE routing mechanism, such as incorrect assignments and imbalanced expert allocation. Inspired by the principles of Redundancy and Fault Tolerance Theory. We innovatively integrate the concept of Mixture of Experts into the routing mechanism and propose an efficient fine-tuning method called Mixture of Routers (MoR). It employs multiple sub-routers for joint selection and uses a learnable main router to determine the weights of the sub-routers. The results show that MoR outperforms baseline models on most tasks, achieving an average performance improvement of 1%. MoR can serve as a plug-and-play, parameter-efficient fine-tuning method suitable for a wide range of applications. Our code is available here: https://anonymous.4open.science/r/MoR-DFC6.

1 Introduction

Large Language Models (LLMs) have gradually become the cornerstone of natural language processing (NLP) (Devlin et al., 2019; Liu et al., 2021; He et al., 2021; Radford et al., 2019). As model parameters increase, LLMs demonstrate impressive emergent abilities and transfer learning capabilities (Wei et al., 2022; Chowdhery et al., 2023; Zhang et al., 2024; Jiang et al., 2024). However, the computational resources required for full fine-tuning are enormous (Lv et al., 2024), and more and more research is focusing on parameter-efficient fine-tuning (PEFT) (Mangrulkar et al., 2022). The main goal is to significantly reduce the resources required for fine-tuning. For example, P-tuning converts prompts into learnable embedding layers (Liu et al., 2022), while LoRA uses a set of low-rank matrices to learn incremental updates (Hu et al., 2022). DCFT (Zhang et al., 2025) further reduces the number of parameters by deconvolution. Despite the high efficiency of PEFT methods, their fine-tuning performance often falls short of meeting the increasingly complex demands of downstream tasks.

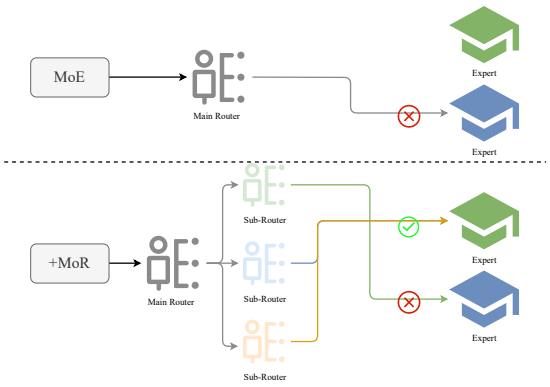

Figure 1: A schematic diagram of the MoR plugin, where the green expert represents the correct expert. MoR effectively corrects misallocations through joint multi-routing assignments.

Mixture-of-Experts (MoE) is designed to improve overall model performance by integrating the advantages of multiple expert networks (Jiang et al., 2024). The core idea of this approach is that different expert networks can specialize in handling different subsets or features of the data, while a gating mechanism is responsible for determining

which expert should process each input (Jacobs et al., 1991; Shazeer et al., 2017a). The model typically includes multiple experts and a gating network. The task of the gating network is to evaluate the relevance of each expert based on the input data and dynamically allocate the input to the expert most suited to handle that data. This dynamic allocation mechanism enables MoE models to be more flexible and efficient, especially when dealing with large-scale and diverse datasets (Lepikhin et al., 2021; Du et al., 2022; Xue et al., 2021; Zuo et al., 2022). Moreover, the MoE architecture offers higher computational efficiency and scalability. By parallelizing the computation across multiple experts, MoE can optimize resource usage, accelerating both training and inference speed. This is particularly important in scenarios that require processing massive datasets.

Recent research has shown that combining PEFT and MoE allows for the advantages of both methods (Zadouri et al., 2024; Liu et al., 2024; Dou et al., 2023). LoRAMoE (Dou et al., 2023) is a plugin version of MoE, learns multiple sets of low-rank matrices as experts and uses a softmax layer as a router to compute the input data for each expert. During training, the pre-trained weights are kept frozen, and only the experts and the router are trained. MoLA (Gao et al., 2024) further investigates the number of experts at different layers, assigning fewer experts to lower layers and increasing the number of experts as the layer depth increases. While it demonstrates impressive results, recent studies have found issues with the MoE routing mechanism, including incorrect assignments and imbalances in expert allocation (Shazeer et al., 2017b; Fedus et al., 2022).

In system design, the principles of Redundancy and Fault Tolerance Theory emphasize the importance of using multiple components to enhance reliability and robustness. By introducing redundancy, systems can mitigate the impact of individual component failures and improve overall performance. Inspired by this theory, we propose a new parameter-efficient MoE method to address the aforementioned issues. Specifically, our approach employs multiple sub-routers for joint decision-making, where each sub-router contributes to the final decision, thereby reducing the risk of errors from any single sub-router. A main router is then used to select the top-r sub-routers based on their scores, ensuring that only the most reliable decisions are prioritized. By adjusting the number of sub-routers, MoR can flexibly adapt to tasks of varying complexity. Ultimately, the weighted cooperation of the sub-routers determines the scores for each expert, and the top-k experts with the highest combined scores are selected for the final inference. We conducted experiments on six benchmarks, including NLP and Commonsense Reasoning (CR) tasks, to demonstrate the effectiveness of MoR. Our main contributions are as follows:

- We propose a new fine-tuning method called MoR, which selects expert models through multiple sub-routers and uses a main router to determine the selection of sub-routers. MoR can replace the router layer in MoE-style models, thereby making it a plug-and-play and parameter-efficient solution.
- We propose a variant of MoR called Consistent Routing Weighting (CRW) to address the impact of sharp, erratic changes in transfer learning, effectively enhancing the model's stability and generalization ability.
- We conduct numerous experiments on Llama2-7B to validate the effectiveness of MoR. We compare it against benchmarks across six different tasks, and MoR outperforms in most of them.

2 Related Work

When performing SFT tasks, full fine-tuning not only requires substantial computational and storage resources but can also lead to catastrophic forgetting. In contrast, PEFT (Mangrulkar et al., 2022) achieves similar results to full fine-tuning by freezing most of the model parameters and training only a small subset of them. Low-Rank Approximation (Hu et al., 2022) is a popular and efficient fine-tuning method for LLMs, dubbed as LoRA. It utilizes low-rank approximation theory to effectively adjust the model's behavior with smaller parameter increments. The forward formula is as follows:

$$W = W^{(0)} + \Delta = W^{(0)} + BA, \qquad (1)$$

where $\Delta \in \mathbb{R}^{din \times dout}$, $A \in \mathbb{R}^{r \times dout}$, and $B \in \mathbb{R}^{din \times r}$, with $r^{\in (din, dout)}$. The dimensions of din and $dout$ are the same as those of the pre-trained matrix W.

Although LoRA significantly reduces the number of parameters, its impact on SFT performance

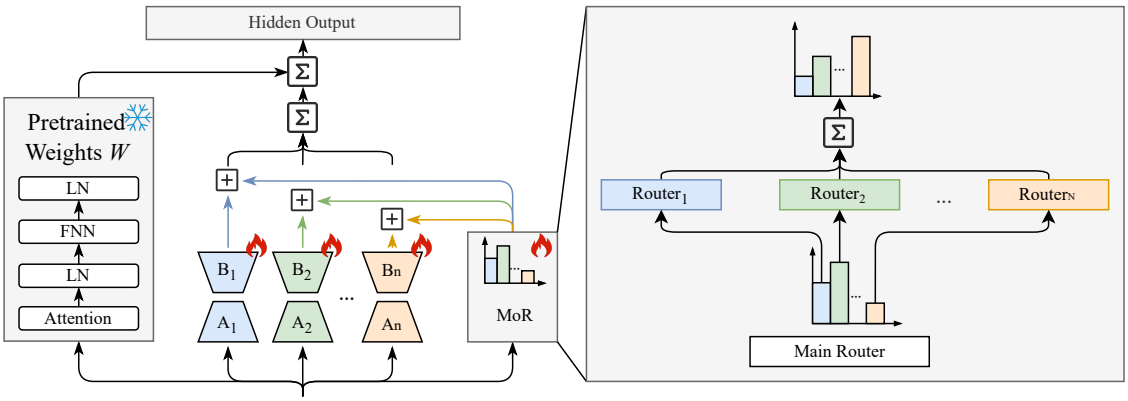

Figure 2: Here is a schematic illustration of MoR. On the left is a schematic diagram of the integration of LoRA and MoE, incorporating LoRA as an expert into the MoE model. On the right is the MoR plugin we propose, which can flexibly replace the router layer in MoE-style models.

is noticeable. The integration of MoE and LoRA represents a notable trend in recent advancements aimed at improving the performance of LLMs degrees. (Zadouri et al., 2024) introduce a novel parameter-efficient MoE framework, Mixture of Vectors (MoV) and Mixture of LoRA (MoLORA), which achieves comparable performance to full fine-tuning with significantly fewer parameters. (Huang et al., 2024) propose LoRAHub, which achieves the cross-task generalization capability of LoRA by integrating LoRA modules trained on different tasks through a simple framework. MoLA (Gao et al., 2024) experimentally demonstrates that more LoRA experts in higher layers can significantly improve the performance of Transformer-based models.

These methods effectively improved the performance of SFT, but they also led to issues such as errors and imbalanced expert allocation (Shazeer et al., 2017b; Fedus et al., 2022). To mitigate the random routing phenomenon observed in MoE, MoELoRA (Luo et al., 2024) encourage experts to learn distinct features through contrastive learning, thereby effectively improving model performance. LoRAMoE (Dou et al., 2023) integrates multiple LoRA experts through a router. It also mitigates the issue of unbalanced expert utilization via a Localized Balancing Constraint. DeepseekV3 (DeepSeek-AI et al., 2024) manages the load through Auxiliary-Loss-Free Load Balancing and Complementary Sequence-Wise Auxiliary Loss. These methods offer valuable insights into errors and imbalanced expert allocation, but our MoR provides a more flexible and plug-and-play strategy for addressing these issues.

3 Method

In this section, we elaborate on the methodological details of MoR. It is a MoE-style plugin that employs multi-route collaborative expert allocation, as illustrated in Figure 2.

3.1 Motivation

In this section, we explore the motivation for introducing multiple routers to make joint decisions and explain their necessity and potential advantages based on the theory of redundancy and fault tolerance.

In traditional MoE-style architectures, a single router is responsible for assigning inputs to the most suitable expert network. However, this design has a significant limitation: the single router may make incorrect allocation decisions due to noise, overfitting, or insufficient training, which can lead to a decline in model performance. This vulnerability is particularly pronounced in high-dimensional and complex tasks. The theory of redundancy and fault tolerance provides important inspiration for addressing this issue. The theory suggests that by introducing redundant components and designing appropriate fault-tolerant mechanisms, the reliability and robustness of a system can be significantly improved. Specifically, when one component in the system fails, other components can collaborate to take over its tasks, thereby preventing overall system failure. Inspired by this, we introduce multiple routers in the MoE architecture to make joint decisions. Aggregating multiple independent judgments reduces the bias of a single decision. If one router makes an incorrect judgment due to noise

or error, the other routers can correct it through collective decision-making, thereby enhancing the system's fault tolerance. Moreover, the design of multiple routers introduces the advantage of diversity. Different routers can form complementary decision-making capabilities through varied initializations, training data, or structural designs. This diversity not only strengthens the system's robustness but also improves the model's adaptability when facing complex tasks.

3.2 Architecture

The left of Figure 2 illustrates the forward process of the standard LoRAMoE architecture. MoE assigns different inputs to different experts via a router module. This means that although adding more experts increases the total number of model parameters, only a small number of experts are involved in the computation during both training and inference. This allows the model parameters to scale with the same computational cost. The router is a trainable gating function that normalizes the distribution of expert weights using the Softmax function. The final output of the MoE layer is the weighted sum of the outputs from the experts:

$$F_i(x) = \frac{\text{Topk}(\text{Softmax}(W_r \cdot x), k)_i}{\sum_{j=1}^{k} \text{Topk}(\text{Softmax}(W_r \cdot x), k)_j}, \quad (2)$$

$$W = W^0 + \sum_{i=1}^{k} F_i(x) \cdot E_i(x), \quad (3)$$

where Topk retains the k highest weight distributions P from the Softmax output, and the remaining weights are set to 0. The retained k weights are then re-normalized to ensure the sum of the weights equals 1. In addition, load balancing loss is commonly used in MoE to promote balanced expert selection. To ensure that the original knowledge space of LLMs is not compromised, LoRA is adopted to reduce the occurrence of catastrophic forgetting. By training multiple pairs of low-rank matrices $\{A\}_{i=1}^{N}$ and $\{B\}_{i=1}^{N}$ as experts. The input is then assigned to different experts through a learnable routing module. Our forward equation is shown as follows:

$$W = W^0 + \sum_{i=1}^{k} F_i(x) \cdot B_i A_i(x). \quad (4)$$

The right of Figure 2 illustrates the MoR plugin. In previous MoE models, experts were selected based on the results of main router W_R, typically choosing the top one or two experts determined by the top-k criterion:

$$R^i = \text{Softmax}(W_R \cdot x)_i, \quad (5)$$

while it improves the coordination ability of expert models in handling complex tasks, it still faces the issues of incorrect and uneven expert allocation. To address the issues, we propose a fine-grained expert control method with a multi-router mechanism, termed MoR, which can be inserted into all MoE-based models by replacing the router component of the original model. It consists of a main router W_R and multiple sub-routers W_r:

$$F_i(x)' = \sum_{i=1}^{N} \frac{R^i}{\sum_{j=1}^{k} R^j} \cdot r^i, \quad (6)$$

where the main router assigns weights to each of the sub-routers. The final expert routing weight R' is a weighted sum of the sub-routers rather than relying on the result of a single router. Specifically, load balancing loss is used in MoR to promote balanced router selection. We summarize the complete training process of MoR in Algorithm 1.

Algorithm 1 MoR Training

Input: Dataset $\mathcal{D}$; total iterations T.
`//Create LoRA experts and routers`
Create matrix $N(A, B)$ and main router W_R and sub-routers $(W_r)_i$;
for $t = 1, \ldots, T$ **do**
  Sample a mini-batch from $\mathcal{D}$;
  `//Calculate the score of experts`
  Compute main router score R as (5);
  Compute the expert score $F_i(x)'$ as (6);
  Select the top-k scoring experts;
  `//Iteratively calculate output` y
  for $i = 1, \ldots, k$ **do**
    $y += B_{(k)} A_{(k)}(x)$;
    return y.
  end for
end for
Output: Expert mixture result y.

3.3 Consistent Routing Weighting

In transfer learning, models face the challenge of distributional differences between the source and target domains. During the process of using MoR for transfer learning, the model heavily relies on the

Models	SciQA	ComQA	OpenQA	MRPC	CoLA	RTE	Avg.
Full-Fine Tuning	93.12	77.48	80.4	87.13	86.29	87.73	85.36
Prompt Tuning	36.78	37.76	46.2	49.91	59.25	54.17	47.35
LLaMA-Adapter	73.33	73.55	71.8	71.94	47.56	72.93	68.52
LoRA	91.01	75.51	77.0	83.13	86.29	85.92	83.14
LoRAMoE	92.04	78.13	80.0	84.23	86.28	85.20	84.31
MoLA	92.36	78.95	79.6	83.48	**86.87**	86.28	84.59
LoRAMoE + MoR	92.90	78.54	81.0	**84.75**	86.39	**88.45**	85.34
MoLA + MoR	**93.08**	**79.20**	**82.0**	83.94	86.77	**88.45**	**85.57**

Table 1: Direct Fine-Tuning Performance comparison of different models on various tasks. We report the accuracy of all the tasks. Higher is better for all metrics. We use the same hyperparameters, which are specified in section C. The best results are denoted in **bold**.

Models	SciQA	ComQA	OpenQA	MRPC	CoLA	RTE	Avg.
LoRA	91.01	74.61	76.6	84.41	**86.95**	84.48	83.01
MoLA	92.94	77.97	78.7	84.52	86.64	86.48	84.54
MoLA + MoR	92.09	77.56	80.0	83.94	85.43	88.09	84.52
MoLA + CRW	**93.21**	**78.79**	**81.6**	**85.39**	86.77	**88.45**	**85.70**

Table 2: Instructional Fine-Tuning Performance comparison of different models on various tasks. CRW is a variant of MoR.

main router, making it difficult to adjust the distribution of expert routes to adapt to new tasks in the early stages of transfer learning. To address these issues, we propose a variant of MoR, called Consistent Routing Weighting (CRW) , to enhance the model's stability and generalization ability. This method ensures parameter stability by applying equal weights across different routes, thereby minimizing the risk of overfitting to specific tasks. By preventing the model from overly relying on features from a particular route, we maintain balanced feature representation, which is crucial when handling tasks from different domains. CRW helps smooth the optimization landscape, making the training process more stable and reducing the likelihood of sharp fluctuations in the loss function. Additionally, CRW enhances cross-domain robustness by ensuring a more uniform contribution from each route, making the model less sensitive to domain-specific variations and better equipped to transfer knowledge effectively. This process can be represented as:

$$F_i(x)^{CRW} = \sum_{i=1}^{N} \frac{1}{N} \cdot r^i. \tag{7}$$

4 Experiments

4.1 Experimental Settings

In this section, we design two experimental setups to evaluate the performance of the MoR method, including direct fine-tuning and instruction fine-tuning. Direct fine-tuning refers to fine-tuning the model directly on downstream tasks, while instruction fine-tuning involves first fine-tuning on an instruction-tuning dataset, followed by fine-tuning on the downstream task. LLaMA-2-7B (Touvron et al., 2023) is used as the base model. To ensure a fair comparison, LoRAMoE allocates 5 experts per layer, while MoLA adopts a similar progressive expert configuration of 2-4-6-8 as mentioned in its paper. We keep the total number of experts the same across all variants. In both settings, we conduct a grid search on the number of training epochs, considering 10, 15, and 20 epochs for fine-tuning on the downstream task. The highest value from the three experiments is taken as the experimental result. We implement all algorithms using PyTorch (Paszke et al., 2019). We use AdamW (Loshchilov and Hutter, 2017) as the optimizer. We applied

Method&# Param	Metric	SciQA	ComQA	OpenQA	MRPC	CoLA	RTE	Avg.
MoLA	Acc(%)	92.36	78.95	79.6	83.48	**86.87**	86.28	84.59
	Time(h)	13.20	19.32	9.67	7.52	15.80	5.25	11.79
MoLA + MoR	Acc(%)	93.08	**79.20**	**82.0**	83.94	86.77	**88.45**	**85.57**
$(r = 2)$	Time(h)	13.70	19.98	10.03	7.92	17.00	5.55	12.36
MoLA + MoR	Acc(%)	**93.30**	78.05	81.2	**84.81**	86.10	87.73	85.20
$(r = 3)$	Time(h)	13.99	20.33	10.27	7.98	17.15	5.55	12.55
MoLA + MoR	Acc(%)	92.99	76.82	80.0	84.00	86.10	88.09	84.67
$(r = 4)$	Time(h)	14.15	20.53	10.57	8.02	17.31	5.58	12.69

Table 3: The results of MoR with different router numbers were tested on 6 different tasks, and we report the accuracy and time efficiency. r represents the number of routers. We use the same hyperparameters, with only the router number varying.

LoRA to four weight matrices in the self-attention module (W_q, W_k, W_v, W_o) and to three weight matrices in the MLP module $(W_{gate}, W_{down}, W_{up})$. All experiments are conducted on a single NVIDIA A100-80G GPU.

4.1.1 Datasets

For evaluation, we adopt three natural language processing (NLP) tasks and three commonsense reasoning (CR) tasks. For the NLP tasks, we evaluate three popular datasets, including Microsoft's Research Paraphrase Corpus (MRPC) (Dolan and Brockett, 2005), Corpus of Linguistic Acceptability (COLA) (Wang et al., 2018), and Recognizing Textual Entailment (RTE) (Wang et al., 2018). For the CR tasks, we evaluate ScienceQA (Lu et al., 2024), CommonsenseQA (Talmor et al., 2019), and OpenbookQA (Mihaylov et al., 2018).

4.1.2 Baselines

We compare MoR with four popular parameter-efficient tuning methods, including prompt tuning (Lester et al., 2021), LLaMAAdapter (Zhang et al., 2024), LoRA (Hu et al., 2022), LoRAMoE (Dou et al., 2023) and MoLA (Gao et al., 2024). Additionally, we also evaluate full-parameter fine-tuning.

4.2 Main Results

Direct Fine-Tuning. We first compare the results of direct fine-tuning between MoR and baseline models on LLAMA-7B, where the accuracy (%) results of MoR and other baselines are shown in Table 1. We use the same hyperparameters for all methods. The results indicate that methods based on LoRA (LoRA, MoLA, and MoR) significantly outperform the baseline methods based on prompt tuning (Prompt Tuning and LLaMA-Adapter). After inserting the MoR module, there is a significant improvement across all tasks. Specifically, MoELoRA and MoLA show increased accuracy rates in the OpenbookQA task by 1.4% and 2.4%, respectively, and in the RTE task by 2.17% and 1.58%, respectively, after the integration of the MoR module. On average, the insertion of the MoR module has led to improvements of 1.03% and 0.98%, respectively. These results demonstrate that MoR can effectively enhance the performance of MoE-style models.

Instructional Fine-Tuning. We first tune LLAMA-7B on the instructional tuning dataset using each PEFT method. Then, we fine-tune for all downstream tasks. Instructional tuning effectively evaluates the transfer learning capabilities of each PEFT method. We only compare methods based on LoRA, as they exhibit stronger transfer learning capabilities compared to methods based on prompt tuning. The results, as shown in Table 2, indicate that MoLA+MoR achieved similar results to MoLA on the Instructional Fine-Tuning task, especially on the ScienceQA task where it decreased by 0.85%. However, after employing the MoR variant CRW, there was a noticeable performance improvement, particularly on the CommonsenseQA and OpenbookQA tasks, where MoLA+CRW improved by 2.9% and 0.82% respectively compared to MoLA. On average, there was a 1.16% improvement. Experimental results show that as a plug-in

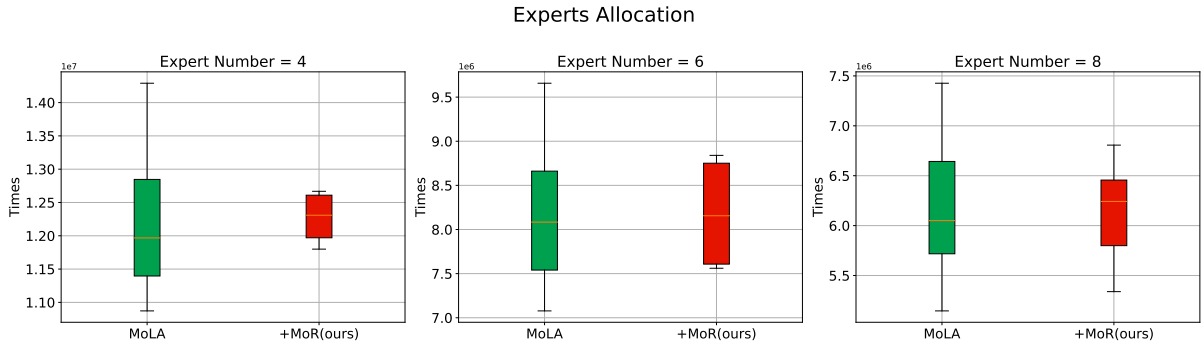

Figure 3: A visualization of expert allocation by MoR and MoLA, showing layers with a total of 4, 6, and 8 experts from left to right. Layers with only 2 experts are not displayed due to the use of the top-2 mechanism. The x-axis represents the expert IDs, and the y-axis represents the number of times each expert is activated.

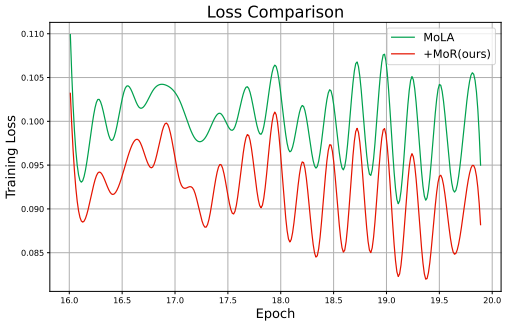

Figure 4: Comparison of training loss between MoR and MoLA on the OpenBookQA task.

for MoE-style models, MoR significantly boosts the performance of direct fine-tuning, while in the Instructional Fine-Tuning task, the MoR variant CRW performs better in terms of transfer learning.

4.3 Analysis

4.3.1 Router Number Analysis

Similar to MoE, MoR can also better adapt to different downstream tasks by adjusting the number of routers. In this section, we present the results and training times of MoR models with different numbers of routers across six distinct tasks, as shown in Table 3. The analysis shows that when using two routers ($r = 2$) for joint allocation, the model achieves the fastest training speed and attains the best overall results, with an average improvement of 0.98% and only a 4.6% increase in training time. When using three routers ($r = 3$) for joint allocation, the model achieves average suboptimal results; however, it is noteworthy that the model produces the best outcomes on the ScienceQA and MRPC tasks. Upon further increasing the number of routers to four ($r = 4$), there is a decline in model performance, along with a 7.1% increase in training time. Further analysis indicates that ScienceQA and MRPC are the most complex tasks among similar types of three tasks, thus requiring more joint allocation experts through routing. In contrast, simpler tasks are prone to overfitting due to excessively high model complexity when using multiple routers.

4.3.2 Expert Allocation Analysis

In this section, we explain the reasons for the performance improvement after inserting the MoR plugin through expert allocation analysis. The changes in expert allocation after inserting MoR are shown in Figure 3. From the figure, we can observe that before inserting MoR, in the left plot, expert(1) is activated excessively frequently, while expert(3) is activated far less often, leading to an imbalanced model workload. Similar activation patterns can also be observed with other numbers of experts. However, after inserting MoR, the distribution of expert activations becomes much more balanced. The difference between the most frequently activated expert and the least frequently activated expert significantly decreases.

Regarding incorrect expert allocations, we can see from the loss curve in Figure 4 that the model with the MoR plugin shows a noticeable reduction in training loss. This indicates that multi-expert routing can better capture underlying patterns in the data, thereby improving the overall performance and generalization ability of the model. Specifically, the MoR plugin introduces a multi-routing mechanism, allowing the model to consider more information when selecting experts. This avoids over-reliance on a single expert or neglecting certain experts. This improvement in load balancing

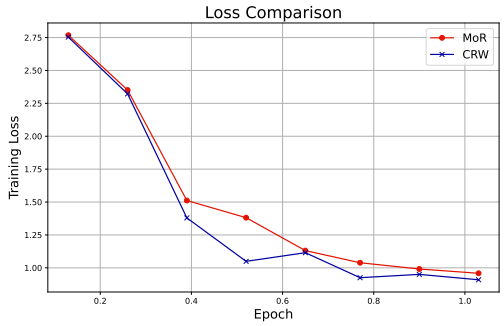

Figure 5: Comparison of training loss between MoR and CRW (a variant of MoR) on the OpenBookQA transfer learning task.

not only enhances training efficiency but also reduces potential biases caused by uneven expert allocation.

4.3.3 Why MoR Performs Poorly on Instruction Fine-tuning Tasks?

MoR dynamically allocates weights via a main router, aiming to flexibly adjust the contribution of each sub-router based on varying task requirements. However, in transfer learning, experimental results show that MoR performs worse than simply using CRW, which contrasts sharply with its superior performance in direct supervised fine-tuning. We recorded the changes in training loss during the early stages of transfer learning, as shown in Figure 5. The results indicate that in the initial phase of transfer learning, CRW exhibits a faster and lower loss reduction. This suggests that equally distributed weights can more efficiently leverage the initial feature representations of multiple routers, allowing for quicker adaptation to the target domain task. In contrast, due to the involvement of the main router, MoR may suffer from suboptimal weight allocation at the beginning, leading to reduced optimization efficiency and slower model convergence. Further analysis reveals that the core reason for this discrepancy lies in the unique characteristics of transfer learning. First, transfer learning faces the challenge of distributional differences between the source and target domains. The main router needs to learn how to allocate weights from limited target domain data. Due to the scarcity of target domain data, the main router is prone to overfitting or developing biases, resulting in suboptimal weight allocation strategies. In comparison, equal weight distribution does not rely on target domain data, avoiding this issue and thus demonstrating stronger robustness. Second, the inherent uncertainty in transfer learning makes it difficult for the main router to accurately capture the importance of each sub-router. If the main router fails to fully understand the characteristics of the target domain, it might mistakenly suppress the contributions of certain critical sub-routers. On the other hand, equal weight distribution ensures equal participation from all sub-routers, fully leveraging the diversity advantage of multi-router architectures.

5 Conclusion

We propose a MoE-style plug-in named MoR, which effectively alleviates the issues of incorrect assignments and imbalances in expert allocation in MoE-style models through multi-routing joint allocation. This innovative, plug-and-play approach provides a flexible solution to the expert allocation problem in MoE models. Extensive experiments conducted on NLP and CR tasks show that MoR outperforms baseline models both in direct fine-tuning and instruction-based fine-tuning scenarios. As a plug-and-play PEFT, MoR can be applied to a wide range of tasks. Moreover, this work offers a promising research direction for enhancing MoE technology and PEFT methods.

Limitations

There are two limitations in this work. First, the current research primarily focuses on experiments with single-modality data and models. Our method has not yet been validated on models handling other modalities. Secondly, due to the lack of computational resources, the experiments in this study were limited to fine-tuning tasks and did not involve the process of training large-scale models from scratch. This means that the performance of our approach in training large models from the ground up has not been fully validated. We look forward to addressing these limitations in the future.

References

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, and et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, pages 1–113.

DeepSeek-AI, Aixin Liu, Bei Feng, and et al. 2024. Deepseek-v3 technical report. *ArXiv*, abs/2412.19437.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and et al. 2019. BERT: Pre-training of deep bidirectional trans-

formers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186.

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

Shihan Dou, Enyu Zhou, Yan Liu, and et al. 2023. Loramoe: Revolutionizing mixture of experts for maintaining world knowledge in language model alignment. *ArXiv*, abs/2312.09979.

Nan Du, Yanping Huang, Andrew M Dai, and et al. 2022. GLaM: Efficient scaling of language models with mixture-of-experts. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 5547–5569.

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

Chongyang Gao, Kezhen Chen, Jinmeng Rao, and et al. 2024. Higher layers need more lora experts. *ArXiv*, abs/2402.08562.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and et al. 2021. Deberta: Decoding-enhanced bert with disentangled attention. In *Proceedings of the International Conference on Learning Representations*.

Edward Hu, Yelong Shen, Phillip Wallis, and et al. 2022. Lora: Low-rank adaptation of large language models. In *Proceedings of the International Conference on Learning Representations*.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, and et al. 2024. Lorahub: Efficient cross-task generalization via dynamic loRA composition. In *First Conference on Language Modeling*.

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

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, and et al. 2024. Mixtral of experts. *ArXiv*, abs/2401.04088.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, and et al. 2021. Gshard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Qidong Liu, Xian Wu, Xiangyu Zhao, and et al. 2024. When moe meets llms: Parameter efficient fine-tuning for multi-task medical applications. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1104–1114.

Xiao Liu, Kaixuan Ji, Yicheng Fu, and et al. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68.

Zhuang Liu, Wayne Lin, Ya Shi, and et al. 2021. A robustly optimized BERT pre-training approach with post-training. In *Proceedings of the Chinese National Conference on Computational Linguistics*, pages 1218–1227.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Pan Lu, Swaroop Mishra, Tony Xia, and et al. 2024. Learn to explain: multimodal reasoning via thought chains for science question answering. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*.

Tongxu Luo, Jiahe Lei, Fangyu Lei, and et al. 2024. Moelora: Contrastive learning guided mixture of experts on parameter-efficient fine-tuning for large language models. *ArXiv*, abs/2402.12851.

Kai Lv, Yuqing Yang, Tengxiao Liu, and et al. 2024. Full parameter fine-tuning for large language models with limited resources. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8187–8198.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

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.

Adam Paszke, Sam Gross, Francisco Massa, and et al. 2019. *PyTorch: an imperative style, high-performance deep learning library*.

Alec Radford, Jeffrey Wu, Rewon Child, and et al. 2019. Language models are unsupervised multitask learners.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, and et al. 2017a. Outrageously

large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, and et al. 2017b. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

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

Hugo Touvron, Louis Martin, Kevin R. Stone, and et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Alex Wang, Amanpreet Singh, Julian Michael, and et al. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Jason Wei, Yi Tay, Rishi Bommasani, and et al. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Fuzhao Xue, Ziji Shi, Futao Wei, and et al. 2021. Go wider instead of deeper. In *AAAI Conference on Artificial Intelligence*.

Ted Zadouri, Ahmet Üstün, Arash Ahmadian, and et al. 2024. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. In *The Twelfth International Conference on Learning Representations*.

Jia-Chen Zhang, Yu-Jie Xiong, Chun-Ming Xia, Dong-Hai Zhu, and Xi-He Qiu. 2025. Parameter-efficient fine-tuning of large language models via deconvolution in subspace. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 3924–3935, Abu Dhabi, UAE. Association for Computational Linguistics.

Renrui Zhang, Jiaming Han, Chris Liu, and et al. 2024. LLaMA-adapter: Efficient fine-tuning of large language models with zero-initialized attention. In *The Twelfth International Conference on Learning Representations*.

Simiao Zuo, Qingru Zhang, Chen Liang, and et al. 2022. MoEBERT: from BERT to mixture-of-experts via importance-guided adaptation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1610–1623.

A Datasets

A.1 NLP Datasets

For evaluation, we adopt the GLUE (General Language Understanding Evaluation) benchmark, which is a widely used collection of datasets designed to assess natural language understanding (NLU) capabilities of models. Specifically, the GLUE benchmark includes several datasets such as COLA (Wang et al., 2018), where the task is to determine whether a given sentence is grammatically acceptable; MRPC (Dolan and Brockett, 2005), which focuses on identifying whether two sentences are semantically equivalent; and RTE (Wang et al., 2018), where the goal is to predict if a premise sentence entails a hypothesis sentence. We present the dataset statistics of GLUE in the following table 4.

Dataset	Metric	#Train	#Valid	#Test	#Label
COLA	Mcc	8.5k	1,043	1,063	2
MRPC	Acc	3.7k	408	1.7k	2
RTE	Acc	2.5k	277	3k	2

Table 4: Dataset Sizes and Evaluation Metrics in the GLUE Benchmark. "Mcc" and "Acc" denote the Matthews correlation coefficient and accuracy

A.2 CR Datasets

Specifically, for the CR tasks, we evaluate ScienceQA (Lu et al., 2024), a dataset that tests multimodal reasoning across text, images, and scientific concepts; CommonsenseQA (Talmor et al., 2019), which focuses on answering questions that require everyday commonsense knowledge and reasoning about implicit relationships; and OpenbookQA (Mihaylov et al., 2018), designed to evaluate reasoning over explicit scientific facts and principles provided in an open book format. For more details and specific statistics, please refer to Table 5.

Dataset	Metric	#Train	#Valid	#Test	#Label
SciQA	Acc	6,508	-	2,224	4
ComQA	Acc	9,740	1,221	-	5
OpenQA	Acc	4,957	500	500	4

Table 5: Dataset Sizes and Evaluation Metrics for Science Question Answering Tasks. "Acc" denotes accuracy.

B Redundancy and Fault Tolerance Theory

Redundancy and Fault Tolerance Theory is a fundamental concept in system design, particularly in engineering, computer science, and reliability analysis. It focuses on enhancing system robustness by incorporating redundant components or mechanisms to ensure continued operation despite failures. The theory is based on the principle that adding extra resources, such as backup systems or duplicate components, can mitigate the impact of faults, thereby improving overall reliability and availability. A key metric in this theory is the system reliability, often modeled using probability:

$$R_{sys} = 1 - (1 - R)^n \tag{8}$$

where R represents the reliability of an individual component, and n is the number of redundant components. This formula illustrates how redundancy increases system reliability exponentially with additional components. Fault tolerance extends beyond hardware to software systems, employing techniques like error detection, correction codes, and failover mechanisms.

C Hyperparameters

To facilitate the reproducibility of our experimental results, we have made the hyperparameters used in our experiments publicly available. This includes detailed configurations such as learning rates, batch sizes, optimization algorithms, weight decay, and other relevant settings. By providing this information, we aim to ensure that our experiments can be replicated and validated by other researchers, fostering transparency and enabling further advancements in the field.

Hyperparameters	Value
batch_size	64
micro_batch_size	4
learning_rate	3e-4
cutoff_len	256
lora_r	8
lora_dropout	0.05
lora_alpha	16

Table 6: Hyperparameters

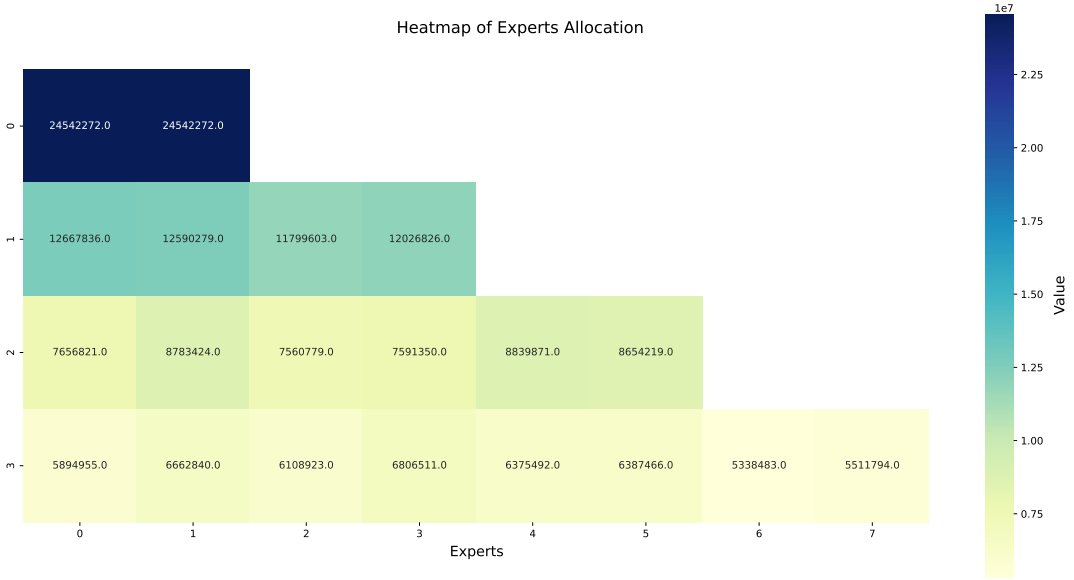

Figure 6: Visualization of MoR expert activations, with the total number of experts being 2, 4, 6, and 8 from top to bottom.

D Visualization of expert allocation

In this section, we present the visualization of expert allocation and activation within the MoR framework, as shown in Figure 6. The similarity in color across each row indicates that the MoR plugin effectively mitigates the issue of improper and uneven expert allocation commonly observed in MoE-style models.