Hybrid Routing for a Mixture of LoRA Experts

Yitong Huang[1,2], **Ziqi Yang**[1,2], **Zihui Wang**[3], **Jianzhong Qi**[4], **Rongshan Yu**[1,2], **Xiaoliang Fan**[1,2*], **Cheng Wang**[1,2]

[1]Fujian Key Laboratory of Urban Intelligent Sensing and Computing, Xiamen University, China
[2]Key Laboratory of Multimedia Trusted Perception and Efficient Computing, Ministry of Education of China, Xiamen University, China
[3]Peng Cheng Laboratory, Shenzhen, China
[4]School of Computing and Information Systems, The University of Melbourne, Australia
huangyitong@stu.xmu.edu.cn, yangziqi@stu.xmu.edu.cn, wangziwei@stu.xmu.edu.cn, jianzhong.qi@unimelb.edu.au, rsyu@xmu.edu.cn, fanxiaoliang@xmu.edu.cn, cwang@xmu.edu.cn

Abstract

Combining Mixture of Experts (MoE) with Low-Rank Adaptation (LoRA) has shown promising efficiency in multi-task instruction tuning for Large Language Models (LLMs). While existing routing schemes for such MoE systems employ auxiliary functions to ensure both expert selection certainty and workload balance among experts, they are hindered by two critical challenges: (1) Existing methods overlook the evolving cross-expert relationships across layers, leading to inefficient expert utilization. (2) The auxiliary functions fail to incorporate cross-task semantic characteristics during expert assignment, leading to suboptimal task adaptation. To address these challenges, we propose **H**ybrid r**out**ing for a **M**ixture **o**f LoRA **E**xperts (**HotMoE**), a novel multi-task instruction tuning framework that adapts hierarchical routing to the distinct characteristics of different LLM layers. First, we design a *hybrid routing module*. In lower layers, expert-expert attention facilitates cross-task collaboration and generalization. In higher layers, token-expert attention enables precise alignment between task semantics and specialized experts. Second, we introduce a *similarity-guided auxiliary loss module* to regularize routing decisions by exploiting hidden state similarities. This loss synergistically reinforces expert specialization without sacrificing certainty of expert selection by promoting cohesive activation patterns among semantically related tasks while sharpening distinctions between conflicting ones. Experiments across two multi-task instruction tuning scenarios covering seven NLP benchmarks demonstrate that HotMoE consistently outperforms all baselines, improving Mean Relative Difference by up to 1.68% with only 3.1% of trainable parameters.

Code — https://github.com/Starlight039/HotMoE

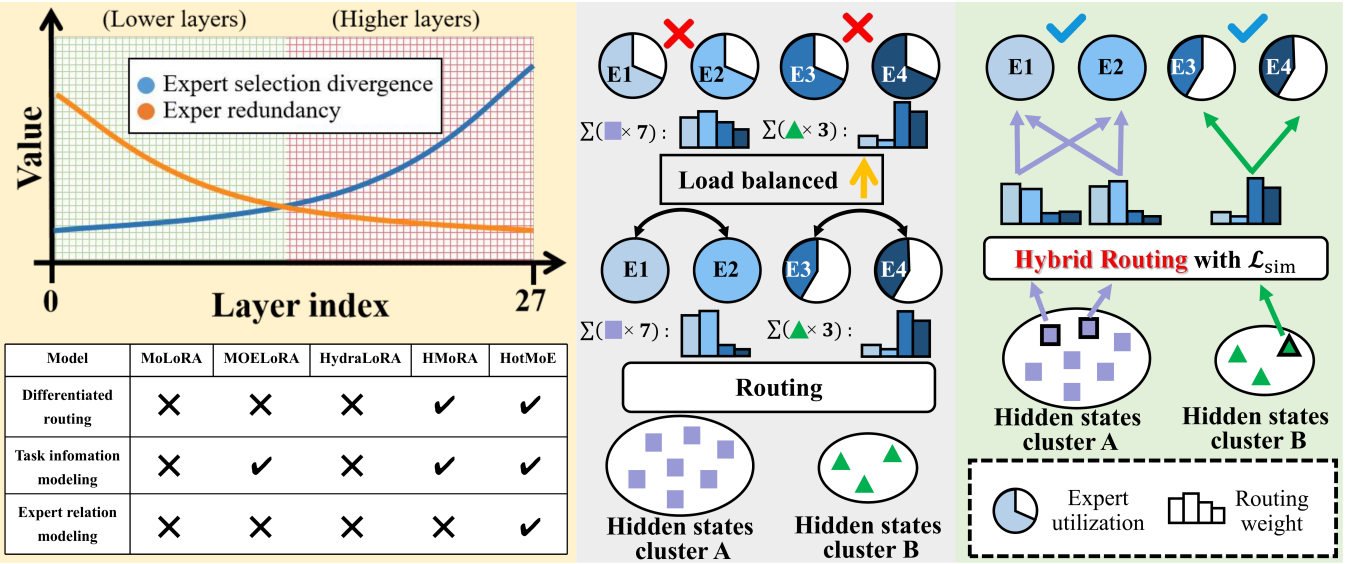
1 Introduction

Instruction tuning has proven effective for enhancing the task-specific capabilities of Large Language Models (LLMs) (Ouyang et al. 2022; Taori et al. 2023; Wei et al. 2022). In practice, LLMs' large number of parameters leads to high computational and memory costs in full fine-tuning. Parameter-efficient fine-tuning (PEFT) methods (Houlsby et al. 2019; Xu et al. 2023) such as LoRA (Hu et al. 2022) address this by freezing pretrained weights and learning low-rank updates, to match full fine-tuning performance at a fraction of the cost. Nevertheless, in multi-task instruction tuning, LoRA's limited capacity exacerbates the seesaw effect—improvements on some tasks come at the expense of the others (Zhao et al. 2024). This stems from misaligned objectives and conflicting gradients across tasks (Ling et al. 2023; Yu et al. 2020), which LoRA struggles to reconcile under tight parameter budgets.

The Mixture of Experts (MoE) architecture introduces sparsely activated parallel expert modules, significantly increasing model capacity with minimal computational overhead (Fedus, Zoph, and Shazeer 2022; Jacobs et al. 1991; Lepikhin et al. 2021; Shazeer et al. 2017). Its input-dependent routing enables dynamic expert assignment, making it well-suited for multi-task learning by mitigating gradient conflicts and task interference (Shen et al. 2024). Building on these strengths, recent works integrate MoE with LoRA, treating LoRA adapters as experts and selecting them via sparse routing. This design incorporates auxiliary functions to ensure load balancing (equal expert usage) and certainty of expert selection (activating only a few relevant experts per input token). Such an architecture improves parameter efficiency and task adaptation at the same time, leading to strong multi-task performance by alleviating the seesaw effect (Feng et al. 2024; Gou et al. 2023; Wang et al. 2024).

However, we observe two challenges in designing an efficient routing scheme that combines MoE with LoRA. **First**, existing methods overlook the variation in expert behavior from lower to higher layers and such uniform routing strategy will lead to inefficient expert utilization. As shown in Figure 1a (top), our preliminary experiments (see Appendix B) reveal a consistent layer-wise pattern: in lower layers, experts tend to be shared across tasks, while in higher layers, expert selection becomes increasingly task-specific. In addition, expert functions evolve from being redundant to becoming more specialized and distinct. This specialization trend is also supported by recent studies (Zhao, Ziser, and Cohen 2024; Gao et al. 2024; Tang et al. 2024), although they do not account for this trend in the design of rout-

*Corresponding author.



Model	MoLoRA	MOELoRA	HydraLoRA	HMoRA	HotMoE
Differentiated routing	✗	✗	✗	✓	✓
Task infomation modeling	✗	✓	✗	✓	✓
Expert relation modeling	✗	✗	✗	✗	✓

(a) (b) (c)

Figure 1: Limitations of the existing routing schemes and intuition of our HotMoE framework. (a) Expert behaviors exhibit significant variations from lower to higher layers, yet such layer-wise difference have not been thoroughly exploited in existing methods (i.e., MoLoRA, MOELORA, HydraLoRA, HMoRA). (b) Current auxiliary losses overlook cross-task semantic relationships, leading to suboptimal expert specialization. (c) Our HotMoE framework could align task semantics to expert capabilities via hybrid routing.

ing mechanisms. To contextualize this observation, we compare design patterns of existing LoRA-MoE methods with our proposed HotMoE in Figure 1a (bottom). Most existing methods (i.e., MoLoRA, MOELoRA, HydraLoRA, and HMoRA) adopting unified routing strategies lack explicit modeling of task information or inter-expert relationships, which limits their ability to exploit the consistent variations across layers. In contrast, our HotMoE explicitly incorporates these dynamics through differentiated routing and task-aware modeling. **Second**, existing auxiliary loss functions for training the MoE routing module result in suboptimal expert specialization and performance degradation. Specifically, they prioritize batch-level load balancing and fail to capture cross-task semantic characteristics. For example, Figure 1b shows two hidden state clusters with distinct task semantics and imbalanced sizes (i.e., 7 states in cluster A, and 3 states in cluster B). Before load balancing, experts E3 and E4 are preferred by the smaller cluster B, indicating lower utilization. Applying balance loss redistributes routing weights, leading to reallocation from E1 and E2 to E3 and E4 for the larger cluster A. This reallocation erroneously reduces routing certainty for states within cluster A and disrupts the specialization of E3 and E4, forcing them to manage tasks with unrelated semantics.

To address those challenges, we propose **HotMoE**, a multi-task instruction tuning MoE framework with a *hybrid routing* scheme. First, in lower layers, we introduce expert-expert attention routing to capture cross-task collaborative relations among LoRA experts, reducing expert redundancy. While in higher layers, we adopt a token-expert attention routing to align fine-grained task semantics with specialized experts, improving expert selection accuracy. Second, to improve the alignment between task semantics and expert capabilities, we further enhance expert specialization and routing certainty by introducing a *similarity-guided auxiliary loss*. This loss function leverages the similarity in hidden states, which reflect task semantics, as a fine-grained proxy for task relations. By promoting consistent activation patterns for related tasks and sharpening distinctions for conflicting ones, this loss could enhance routing accuracy and reduces task interference. Third, to better reflect performance trade-offs across tasks with varying difficulty and resource levels, we introduce Mean Relative Difference (MRD) as a complementary evaluation metric. Extensive experiments on two multi-task instruction tuning scenarios covering seven NLP benchmarks demonstrate that HotMoE outperforms all baselines up to 1.68% improvement in MRD with 3.1% trainable parameters. Our contributions are summarized as follows:

- We propose HotMoE, a novel multi-task instruction tuning framework that integrates a hybrid routing scheme with similarity-guided auxiliary loss to enable effective task-expert alignment and expert specialization.
- We design a hybrid routing strategy where expert–expert attention routing in lower layers fosters expert collaboration, and token–expert attention routing in higher layers aligns task-specific semantics with expert capabilities.
- Extensive experiments on multi-task instruction tuning benchmarks with multiple metrics (including the newly defined MRD) show that HotMoE effectively mitigates task conflicts and improves overall performance.

2 Related Work

Mixture of Experts. The MoE architecture (Jacobs et al. 1991) comprises multiple expert subnetworks and a router that assigns experts for each input. Shazeer et al. (2017) propose sparse routing to activate only a subset of experts per input, thereby boosting computational efficiency and model capacity. Gupta et al. (2022) demonstrate that MoE's sparse activation mechanism reduces task interference and delivers stronger robustness and generalization in multi-task joint training. Some approaches enhance MoE routing using task labels. Ye, Zha, and Ren (2022) enhance MoE by incorporating a task-level router, while Aoki, Tung, and Oliveira (2022) reduce conflicts by assigning independent routers to each task. Our approach differs from these in that we capture token-level similarity to enable semantically coherent routing beyond fixed task labels.

Mixture of LoRA Experts. Recent works treat parallel LoRA modules as experts, integrating LoRA with MoE to enhance multi-task capability while preserving parameter efficiency. Several studies construct MoE architectures by combining independently trained LoRA for different tasks (Feng et al. 2024; Wu, Huang, and Wei 2024; Xu, Lai, and Huang 2025). Other studies train LoRA experts from scratch. Liu et al. (2024) introduce a task-aware gating function to conduct multi-task learning in the medical domain. Zhao et al. (2025) treat LoRA with different ranks as independent experts. Liu and Luo (2024) dynamically adjust the number of activated experts based on task complexity. Luo et al. (2024) leverage contrastive learning to encourage each expert to learn distinct representations. Most of these methods rely on the classic routing scheme that overlooks the correlations among experts. In contrast, we incorporate attention mechanisms to model expert correlations and guide more accurate routing decisions.

3 Methodology

3.1 Preliminaries

Multi-Task Instruction Tuning. We consider a multi-task instruction tuning scenario to tune a pre-trained LLM $\mathcal{M}$ for C tasks $\{T_i\}_{i=1}^{C}$, where each task differs in input distributions and learning objectives, e.g., commonsense reasoning, reading comprehension, translation, and natural language inference. All tasks are jointly trained under a unified parameter space, each sample consists of three components: input instruction, task-specific context, and output target. The overall objective is to generalize model $\mathcal{M}$ towards all C tasks by promoting efficient parameter sharing and task decoupling, while alleviating the negative transfer commonly observed in multi-task learning.

Mixture of LoRA Experts Architecture. Inspired by GShard (Lepikhin et al. 2021), we insert LoRA experts into the Feed-Forward Network (FFN) layers of pretrained Transformer blocks. These layers contain most of the trainable parameters. Such a design allows our fine-tuned model to better specialize across heterogeneous tasks by enabling expert diversity in the most expressive parts of the model. Given a pretrained linear layer with a weight matrix $\mathbf{W}_p \in \mathbb{R}^{d_{\text{input}} \times d_{\text{output}}}$, where d_{input} and d_{output} denote the input and output dimensionalities of the layer, respectively, we define each expert as a pair of low-rank projection matrices:

$$\{(\mathbf{W}_{A_i}, \mathbf{W}_{B_i})\}_{i=1}^{N}, \tag{1}$$

where $\mathbf{W}_{A_i} \in \mathbb{R}^{d_{\text{input}} \times r}, \mathbf{W}_{B_i} \in \mathbb{R}^{r \times d_{\text{output}}}$, N is the total number of experts and r is the rank of LoRA. To determine which experts to be activated for inputs, we use a routing function $\phi : \mathbb{R}^{d_{\text{input}}} \to \mathbb{R}^N$, which produces a routing weight vector $\mathbf{z} \in \mathbb{R}^N$ for each hidden state $\mathbf{h} \in \mathbb{R}^{d_{\text{input}}}$. To reduce inference overhead, we employ top-k routing, activating only the k experts with the highest routing probabilities. The output of the mixture of LoRA experts is computed as:

$$\mathbf{z} = \text{Softmax}(\phi(\mathbf{h})), \tag{2}$$

$$\hat{\mathbf{h}} = \sum_{i=1}^{K} \text{TopK}(\mathbf{z})_i \cdot ((\mathbf{h}\mathbf{W}_{A_i})\mathbf{W}_{B_i}), \tag{3}$$

where $\hat{\mathbf{h}} \in \mathbb{R}^{d_{\text{output}}}$. During training, only the LoRA and routing parameters are updated, while the pretrained backbone weights of $\mathcal{M}$ are frozen, enabling parameter-efficient task adaptation through conditional computation.

3.2 The HotMoE Framework

As shown in Figure 2, HotMoE adapts mixture of LoRA experts and enhances multi-task learning through two key modules: (1) **Hybrid routing** with different routing strategies across lower and higher layers of the target model $\mathcal{M}$. (2) **Similarity-guided auxiliary loss** is applied at all layers during training to regularize routing behaviors, using routing weights and hidden states as inputs. We will introduce those two modules in detail as follows.

3.3 Hybrid Routing Module

To efficiently utilize experts, we design a hybrid routing module where expert–expert attention in lower layers promotes cross-task collaboration, while token–expert attention in higher layers ensures precise alignment between task semantics and specialized experts. Given a model $\mathcal{M}$ with L layers, layers 0 to $r_1 \cdot L$ (exclusive) are defined as the lower layers, and the remaining ones as the higher layers, where r_1 is a hyperparameter. We analyze different r_1 values in Appendix E.2, and find that a balanced routing ratio achieves a good trade-off between collaboration and specialization, adapting well across diverse multi-task settings.

Lower Layers: Expert-Expert Attention Routing. Inspired by Wu et al. (2024), to uncover the cross-expert correlations through attention, we introduce layer-wise attention-based routing modules that explicitly model expert correlations while enabling expert selection to appropriately reflect the task-specific preferences across layers. In lower layers where token representations tend to be semantically task-agnostic, we employ an expert-expert attention routing strategy to promote cross-task generalization. We use a set of **global expert representations** $\mathbf{E}_{\text{global}}$ shared across inputs from different tasks, constructing a unified expert relationship space that does not vary given different inputs.

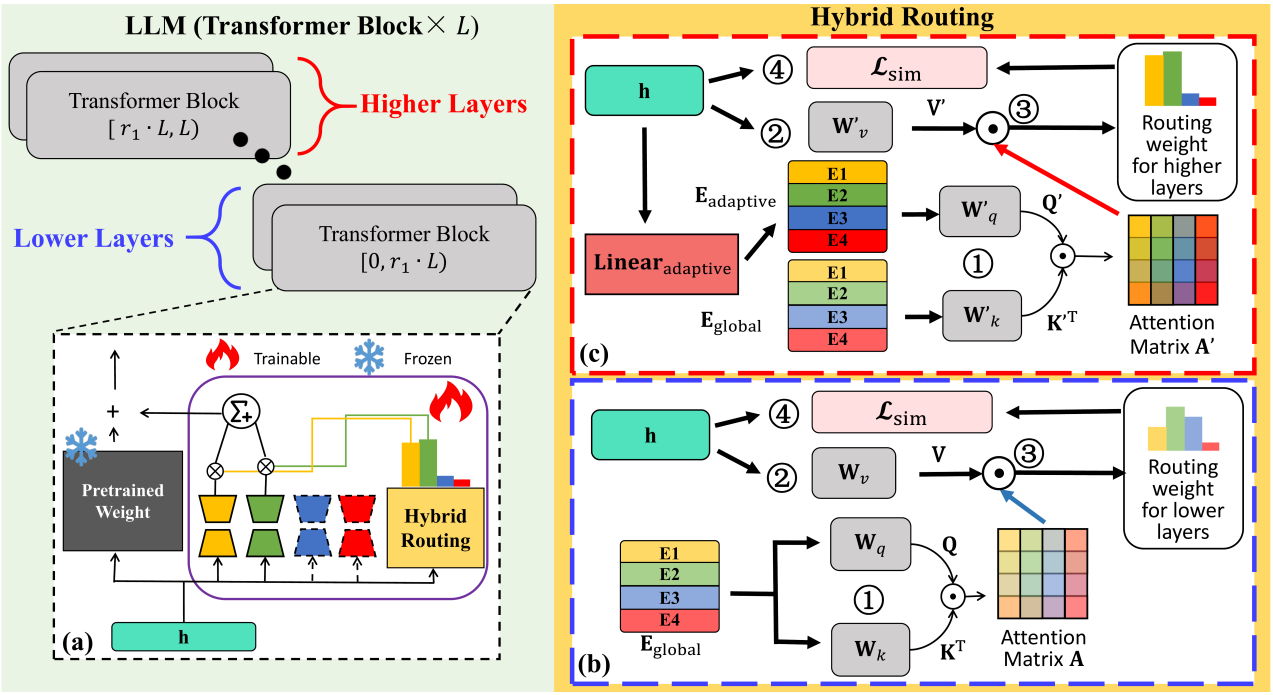


Figure 2: The HotMoE framework. (a) Mixture of LoRA experts architecture. (b) Expert-expert attention routing for lower layers in **hybrid routing**. (c) Token-expert attention routing for higher layers in **hybrid routing**. Workflow steps are as follows: ① The expert-expert routing computes attention among global expert representations $\mathbf{E}_{\text{global}}$. The token-expert routing computes attention between $\mathbf{E}_{\text{global}}$ and input-adaptive expert representations $\mathbf{E}_{\text{adaptive}}$. ② A hidden state $\mathbf{h}$ is projected to produce initial routing logit. ③ The generated attention matrices are used to weight the routing logits to produce routing weights reflecting expert correlations. ④ The **similarity-guided auxiliary loss** $\mathcal{L}_{\text{sim}}$ is computed on the routing weights and hidden states.

$\mathbf{E}_{\text{global}} \in \mathbb{R}^{N \times d_{\text{expert}}}$ is parameterized as a trainable embedding matrix, where each row corresponds to a learnable expert embedding, and d_{expert} is the dimensionality. By computing pairwise attention scores among these embeddings, we obtain a learnable coefficient matrix that captures expert dependencies. This matrix is then used to refine routing weights, ensuring consistent interactions among experts.

The routing process consists of three main steps. First, to capture the relationships among experts, we perform the attention operation over $\mathbf{E}_{\text{global}}$. These expert embeddings are projected into query and key spaces:

$$\mathbf{Q} = \text{LayerNorm}(\mathbf{E}_{\text{global}})\mathbf{W}_q, \tag{4}$$

$$\mathbf{K} = \text{LayerNorm}(\mathbf{E}_{\text{global}})\mathbf{W}_k, \tag{5}$$

where $\mathbf{W}_q, \mathbf{W}_k \in \mathbb{R}^{d_{\text{expert}} \times d_{\text{router}}}$, $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{N \times d_{\text{router}}}$, and d_{router} denotes the dimensionality of the attention projection in the router. LayerNorm (Ba, Kiros, and Hinton 2016) is applied to stabilize training and improve the effectiveness of attention computation. The attention matrix is computed as:

$$\mathbf{A} = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_{\text{router}}}}\right), \tag{6}$$

where $\mathbf{A} \in \mathbb{R}^{N \times N}$ and $\mathbf{A}_{i,j}$ captures the attention from expert i to expert j. Second, each hidden state $\mathbf{h}$ is projected into the value space through a linear transformation to obtain initial routing logits:

$$\mathbf{V} = \mathbf{h}\mathbf{W}_v, \tag{7}$$

where $\mathbf{W}_v \in \mathbb{R}^{d_{\text{input}} \times N}$ is learnable, and $\mathbf{V} \in \mathbb{R}^N$. Finally, the expert attention matrix is used to refine the initial routing logits:

$$\mathbf{z} = \text{Softmax}(\mathbf{A}\mathbf{V}), \tag{8}$$

where $\mathbf{z} \in \mathbb{R}^N$ is the final routing weight enriched with cross-expert dependencies via attention.

This design facilitates the extraction of generalizable features in the lower layers, forming a solid foundation for learning task-specific semantics in the higher layers.

Higher Layers: Token-Expert Attention Routing. In higher layers, the learned representations of input tokens are increasingly task-dependent. While $\mathbf{E}_{\text{global}}$ establishes foundational expert features, additional mechanisms are required to resolve the fine-grained preference divergences towards different tasks. We propose the token-expert attention routing, which introduces **input-adaptive expert representations** $\mathbf{E}_{\text{adaptive}}$ for each hidden state. Generated by a trainable network, $\mathbf{E}_{\text{adaptive}}$ reflects the hidden state's affinity towards different expert capabilities. This allows the routing function to dynamically adapt the routing weights based on task semantics of each hidden state. By combining $\mathbf{E}_{\text{global}}$ and $\mathbf{E}_{\text{adaptive}}$ through attention, we achieve fine-grained expert selection that aligns each input with its most suitable experts, thereby accelerating expert specialization in task-sensitive layers. First, the hidden state is passed through a

network $\text{Linear}_{\text{adaptive}} : \mathbb{R}^{d_{\text{input}}} \to \mathbb{R}^{N \times d_{\text{expert}}}$:

$$\mathbf{E}_{\text{adaptive}} = \text{Linear}_{\text{adaptive}}(\mathbf{h}), \tag{9}$$

where each row of $\mathbf{E}_{\text{adaptive}} \in \mathbb{R}^{N \times d_{\text{expert}}}$ serves as a dynamic "expert capability preference" representation for the hidden state, capturing how each expert aligns with hidden state's task semantic. To model task–expert affinities, we apply attention between $\mathbf{E}_{\text{adaptive}}$ and $\mathbf{E}_{\text{global}}$:

$$\mathbf{Q}' = \text{LayerNorm}(\mathbf{E}_{\text{adaptive}})\, \mathbf{W}'_q, \tag{10}$$

$$\mathbf{K}' = \text{LayerNorm}(\mathbf{E}_{\text{global}})\, \mathbf{W}'_k, \tag{11}$$

$$\mathbf{A}' = \text{Softmax}\left(\frac{\mathbf{Q}'\mathbf{K}'^{\top}}{\sqrt{d_{\text{router}}}}\right), \tag{12}$$

where $\mathbf{W}'_q, \mathbf{W}'_k \in \mathbb{R}^{d_{\text{expert}} \times d_{\text{router}}}$, $\mathbf{Q}', \mathbf{K}' \in \mathbb{R}^{N \times d_{\text{router}}}$ and $\mathbf{A}' \in \mathbb{R}^{N \times N}$ captures input-conditioned affinities among experts. Then, $\mathbf{h}$ is projected into initial routing logits via a linear transformation:

$$\mathbf{V}' = \mathbf{h}\mathbf{W}'_v, \tag{13}$$

where $\mathbf{W}'_v \in \mathbb{R}^{d_{\text{input}} \times N}$ and $\mathbf{V}' \in \mathbb{R}^{N}$. Finally, we use $\mathbf{A}'$ to refine the initial routing logits as:

$$\mathbf{z}' = \text{Softmax}(\mathbf{A}'\,\mathbf{V}'), \tag{14}$$

yielding routing weights that integrate both static expert features and input-adaptive preferences.

Both $\mathbf{E}_{\text{global}}$ and $\mathbf{E}_{\text{adaptive}}$ learn expert preference during training, capturing how each expert tends to respond to specific input patterns, and the attention mechanism estimates the compatibility between these learned preferences. This design fosters the co-activation of experts with similar input affinities, thereby encouraging collaboration among experts with semantically aligned capabilities. Simultaneously, it also enables the functional specialization of experts that have distinct and divergent preferences.

3.4 Similarity-Guided Auxiliary Loss Module

In MoE with sparse routing, prior works often adopt entropy-based objectives to promote both balanced expert usage and confident expert selection (Chen et al. 2023; Liao et al. 2025; Shen et al. 2024). A typical formulation is the Generalized Jensen–Shannon Divergence (GJS) (Lin 1991). A detailed formulation of GJS can be found in Appendix A. While maximizing GJS jointly promotes load balance and certainty of expert selection, these objectives overlook the semantic relationships among inputs. Since hidden states inherently encode task semantics, similar hidden states may arise from different tasks that share analogous intents or structures, forming semantic clusters. Hidden state similarity provides fine-grained signals of cross-task relationships, as semantically clustered inputs often share consistent routing patterns. When clusters vary in size, enforcing global balance ignoring these relationships may lead to expert interference across unrelated clusters, undermining routing certainty and expert specialization.

We propose a similarity-guided auxiliary loss that leverages the semantic structure of hidden states to provide fine-grained guidance for expert selection. This loss function comprises two complementary objectives: (1) $\mathcal{L}_{\text{cons}}$ encourages inputs with similar semantic features to activate similar experts; (2) $\mathcal{L}_{\text{div}}$ encourages semantically dissimilar inputs to activate different experts.

$$\mathcal{L}_{\text{cons}} = \frac{1}{M^2} \sum_{i=1}^{M} \sum_{j=1}^{M} \left(1 - \cos(\mathbf{z}_i, \mathbf{z}_j)\right) \cos(\mathbf{h}_i, \mathbf{h}_j), \tag{15}$$

$$\mathcal{L}_{\text{div}} = \frac{1}{M^2} \sum_{i=1}^{M} \sum_{j=1}^{M} \cos(\mathbf{z}_i, \mathbf{z}_j)\, \left(1 - \cos(\mathbf{h}_i, \mathbf{h}_j)\right), \tag{16}$$

where M denotes the batch size, $\cos(\mathbf{h}_i, \mathbf{h}_j)$ denotes the cosine similarity between hidden states $\mathbf{h}_i$ and $\mathbf{h}_j$, and $\cos(\mathbf{z}_i, \mathbf{z}_j)$ measures the similarity between their routing distributions. The combined similarity-guided loss is:

$$\mathcal{L}_{\text{sim}} = \beta_c\, \mathcal{L}_{\text{cons}} + \beta_d\, \mathcal{L}_{\text{div}}, \tag{17}$$

where β_c and β_d are hyperparameters controlling the strength of each term. We combine the negative GJS with $\mathcal{L}_{\text{sim}}$ to optimize load balance and routing certainty, while considering task semantic relationships. The new auxiliary loss is defined as:

$$\mathcal{L}_{\text{aux}} = -\text{GJS} + \mathcal{L}_{\text{sim}}. \tag{18}$$

Thus, the overall training objective becomes:

$$\mathcal{L} = \mathcal{L}_{\text{model}} + \gamma\, \overline{\mathcal{L}_{\text{aux}}}, \tag{19}$$

where $\mathcal{L}_{\text{model}}$ is the language modeling loss of $\mathcal{M}$, γ is a hyperparameter, and $\overline{\mathcal{L}_{\text{aux}}}$ is the average $\mathcal{L}_{\text{aux}}$ of all routers.

From the clustering perspective, the similarity-guided loss delivers several key benefits: (1) **Fine-grained consistency**: hidden states within the same cluster are encouraged to share experts, reinforcing those experts' capabilities to extract common semantic features and forming stable, specialized expert groups. Cluster-consistent routing promotes expert collaboration on similar task-agnostic features, aligning well with expert-expert routing in lower layers. (2) **Cluster-level differentiation**: different clusters are encouraged to activate different experts, sharpening semantic boundaries and reducing task interference. Cluster-varying routing drives experts to focus on task-specific semantics, accelerating expert specialization and effective task decomposition, which suits token-expert routing in higher layers. (3) **Mitigating side-effects of expert load balancing**: when clusters vary in size, enforcing strict load balancing can undermine expert specialization. Our hidden states similarity-guided loss respects cluster size imbalances, encouraging consistent routing within each cluster while maintaining cluster separation, thereby preventing over-balanced expert usage.

In conclusion, by tolerating a slight deviation in expert balance, $\mathcal{L}_{\text{sim}}$ facilitates more semantically coherent expert assignments and achieves improved routing certainty. Such a trade-off appears reasonable in scenarios where task boundaries are well-defined. The potential impact of the proposed $\mathcal{L}_{\text{sim}}$ on load balancing is analyzed in Appendix E.3.

4 Experiments

4.1 Mean Relative Difference

In multi-task learning scenarios, using average performance (denoted as AVG) across tasks as the sole evaluation metric

Model	Param	MRPC	RTE	Swag	Comm	WMT19	AVG	MRD
LoRA single (r = 8)	0.46%	74.33	79.50	66.17	33.78	19.62	54.68	-
LoRA mix (r = 8)	0.46%	75.50	78.67	65.00	34.89	19.45	54.70	+0.24%
LoRA mix (r = 40)	2.28%	77.17	81.33	73.33	32.28	16.53	56.13	-0.65%
MoLORA	2.39%	70.17	80.50	65.67	35.44	19.16	54.19	-0.50%
MOELoRA	2.28%	75.00	78.67	61.17	36.65	19.61	54.22	+0.15%
HydraLoRA	1.72%	75.33	80.67	65.83	34.81	18.69	55.07	+0.12%
HMoRA	3.63%	74.67	78.83	62.67	34.49	19.62	54.06	-0.71%
HotMoE (ours)	3.19%	73.17	80.67	67.67	35.64	19.75	55.38	**+1.68%**

Table 1: Performance under the task synergy scenario. **AVG** denotes model performance averaged over all tasks. **Bold** indicates the best MRD result, underline denotes the second-best MRD result, and results with a gray background indicate better performance than model $\mathcal{M}$ fine-tuned with LoRA for each task separately. All results (larger is better) are averaged over three runs. Here, **Comm** stands for **CommonGen** and **Swag** stands for **HellaSwag**.

Model	Param	ARC-C	RTE	BoolQ	Comm	WMT19	AVG	MRD
LoRA single (r = 8)	0.46%	66.17	79.50	68.67	33.78	19.62	53.55	-
LoRA mix (r = 8)	0.46%	63.33	76.50	70.33	34.20	18.67	52.61	-1.85%
LoRA mix (r = 40)	2.28%	63.17	79.67	75.00	31.39	17.26	53.30	-2.84%
MoLORA	2.39%	62.50	79.33	68.17	33.02	18.64	52.33	-2.75%
MOELoRA	2.28%	60.83	80.00	71.17	36.22	18.24	53.29	-0.72%
HydraLoRA	1.72%	68.17	77.67	73.83	33.27	17.83	54.15	-0.49%
HMoRA	3.63%	59.33	79.50	67.17	36.43	20.40	52.57	-0.14%
HotMoE (ours)	3.19%	65.33	81.67	70.33	33.84	19.07	54.05	**+0.25%**

Table 2: Performance under the task conflict scenario (same result presentation style as in Table 1).

can be misleading. Due to the seesaw effect, models may exhibit a higher performance score on average simply by overfitting easier or high-resource tasks, even if they have not performed as well on harder or low-resource tasks. To better reflect this phenomenon, we introduce **Mean Relative Difference (MRD)** as a novel evaluation metric.

Let $\{D_i\}_{i=1}^{C}$ be the collection of datasets for multi-task learning. We first fine-tune model $\mathcal{M}$ on each D_i independently using LoRA and record the corresponding single-task test performance as $score_i$. Then, for each multi-task method under evaluation, we let mix_score_i be the test performance of the multi-task model on task i and compare it with the $score_i$. The MRD is calculated as the relative performance gain averaged over the C tasks:

$$\text{MRD} = \frac{1}{C}\sum_{i=1}^{C}\frac{mix\_score_i - score_i}{score_i}. \tag{20}$$

By combining MRD with traditional metrics such as AVG, we gain a more comprehensive understanding of model performance over diverse tasks.

To test how our fine-tuned model performs in multi-task learning, we construct two representative scenarios: (1) In the **task synergy scenario**, tasks, e.g., MRPC (paraphrase detection) and RTE (natural language inference), share similar semantic understanding objectives. They benefit from joint training, and multi-task LoRA training could lead to a **positive MRD**. (2) In the **task conflict scenario**, the multiple tasks interfere with each other, e.g., ARC-C (question answering) and BoolQ (reading comprehension), which require distinct reasoning strategies. In this scenario, joint training is likely to lead to a **negative MRD** and is more difficult to achieve a positive MRD.

4.2 Experimental Setup

Datasets. We conduct experiments on a diverse set of NLP tasks, all converted into instruction tuning formats. These tasks include: **MRPC** (Dolan and Brockett 2005), **HellaSwag** (Zellers et al. 2019), **RTE** (Dagan et al. 2022), **ARC-C** (Clark et al. 2018), **BoolQ** (Clark et al. 2019), the English to Chinese translation subset of **WMT19** (Junczys-Dowmunt 2019), and **CommonGen** (Lin et al. 2020). Details about these datasets are included in Appendix D.1.

Metrics. For classification tasks (MRPC, HellaSwag, RTE, ARC-C, and BoolQ), we use accuracy as the evaluation metric. For the machine translation task (WMT19), we use BLEU (Papineni et al. 2002), and for the generation task (CommonGen), we use ROUGE-L (Lin and Och 2004). We also report MRD, as defined in Section 4.1, to evaluate the performance under **task synergy scenario** (MRPC, RTE, HellaSwag, CommonGen, WMT19) and **task conflict scenario** (ARC-C, RTE, BoolQ, CommonGen, WMT19).

Baselines. We use Qwen2-1.5B (Yang et al. 2024) as the base model to conduct multi-task experiments, and compare our framework with the following representative baselines: **LoRA** (Hu et al. 2022), **MoLORA** (Zadouri et al. 2024), **MOELoRA** (Liu et al. 2024), **HydraLoRA** (Tian et al. 2024), and **HMoRA** (Liao et al. 2025). For **LoRA single** ($r = 8$), we fine-tune $\mathcal{M}$ with LoRA and test it on each task separately. Its performance scores serve as $score_i$ in MRD calculation. **LoRA mix** and other baselines are trained

Model	ARC-C	RTE	BoolQ	Comm	WMT19	AVG	MRD
HotMoE	**65.33**	**81.67**	**70.33**	33.84	19.07	**54.05**	**+0.25%**
HotMoE-$\mathbf{W}_g$	63.67	80.17	68.50	33.64	**19.82**	53.16	-0.52%
HotMoE-$\mathbf{W}_g$ (lower)	64.67	79.33	70.17	**35.26**	18.79	53.64	-0.03%
HotMoE-$\mathbf{W}_g$ (higher)	64.83	80.50	69.50	34.39	18.92	53.63	-0.26%

Table 3: Ablation study of routing scheme on task conflict scenario. **Bold** indicates the best result in each column.

Model	ARC-C	RTE	BoolQ	Comm	WMT19	AVG	MRD
HotMoE w/o $-$GJS &$\mathcal{L}_{\text{sim}}$	63.33	77.17	**70.67**	<u>34.14</u>	<u>19.31</u>	52.92	-0.96%
HotMoE w/o $\mathcal{L}_{\text{sim}}$	<u>63.50</u>	<u>79.83</u>	70.17	33.59	19.18	<u>53.25</u>	-0.85%
HotMoE w/o $-$GJS	62.67	79.33	69.67	**34.83**	**19.37**	53.17	<u>-0.44%</u>
HotMoE	**65.33**	**81.67**	<u>70.33</u>	33.84	19.07	**54.05**	**+0.25%**

Table 4: Ablation study of auxiliary loss on task conflict scenario. **Bold** indicates the best result, and <u>underline</u> denotes the second-best.

jointly on the mixed dataset following a multi-task learning paradigm. Baseline overviews are provided in Appendix C.

Implementation Details. For all MoE-LoRA models, we use 5 experts, each with a rank $r = 8$. A top-2 routing strategy is adopted in MOELoRA, HMoRA, and HotMoE to enable sparse expert activation. The auxiliary loss $\mathcal{L}_{\text{aux}}$ is configured with hyperparameters $\beta_c = 1.0$, $\beta_d = 1.5$, and $\gamma = 0.01$. For hybrid routing, we set the ratio r_1 to 0.5. Both d_{expert} and d_{router} are fixed at 256 across all layers. More implementation details are provided in Appendix D.

4.3 Overall Results

Tables 1 and 2 report the overall performance comparison results on the two testing scenarios.

In Table 1, when collaborative tasks are jointly trained with a fixed rank of 8, LoRA mix ($r = 8$) consistently outperforms LoRA single ($r = 8$). Both the performance gain on AVG and the positive MRD indicate that joint training successfully captures cross-task commonalities. Although increasing the rank to 40, LoRA mix ($r = 40$), results in the highest AVG, it overfits to easier tasks like MRPC, RTE, and HellaSwag, at the expense of harder tasks such as CommonGen and WMT19. Its negative MRD suggests that simply scaling parameters cannot enhance performance on each task. In contrast, our HotMoE enhances performance on multiple tasks with minimal performance drops on MRPC. HotMoE also outperforms existing MoE-LoRA models in metrics of AVG and MRD.

As shown in Table 2, our HotMoE is the only method that achieves a positive MRD in the task conflict scenario. LoRA mix ($r = 40$) still suffers from seesaw effects, while sparse-routing MoE variants mitigate such conflicts by allowing each task to activate only a subset of experts rather than enforcing full expert sharing. Moreover, HotMoE achieves a higher AVG, further demonstrating its robustness in complex and conflicting task environments.

Model	$\mathcal{L}_{\text{aux}}$	MRPC	RTE	Swag	Comm	WMT19	AVG	MRD
HotMoE-$\mathbf{W}_g$	-	68.50	**81.17**	**67.17**	33.36	19.12	53.87	-1.60%
HotMoE-$\mathbf{W}_g$	-GJS	**71.67**	80.83	66.17	32.72	19.34	54.15	-1.29%
HotMoE-$\mathbf{W}_g$	-GJS+$\mathcal{L}_{\text{sim}}$	71.33	80.50	66.33	**33.44**	**19.54**	**54.23**	**-0.79%**
MoLORA	-	70.17	**80.50**	65.67	**35.44**	19.16	54.19	-0.50%
MoLORA	-GJS	72.33	78.67	65.83	34.81	**19.53**	54.23	-0.33%
MoLORA	-GJS+$\mathcal{L}_{\text{sim}}$	**73.67**	80.33	**67.17**	33.79	19.35	**54.86**	**+0.06%**

Table 5: Performance comparison for the $\mathbf{W}_g$ router or MoLORA under different $\mathcal{L}_{\text{aux}}$ for the task synergy scenario. **Bold** indicates the best result for each model.

4.4 Ablation Study

To study the contribution of each module in HotMoE, we conduct two ablation experiments under the task conflict scenario. First, to evaluate the effectiveness of the proposed hybrid routing, we replace it with a standard router $\mathbf{W}_g \in \mathbb{R}^{d_{\text{input}} \times N}$ equipped with top-2 routing. This variant is denoted as HotMoE-$\mathbf{W}_g$. We further isolate the contribution of each part in our hierarchical design by replacing only the lower layers' expert-expert routing or the higher layers' token-expert routing with $\mathbf{W}_g$, resulting in HotMoE-$\mathbf{W}_g$ (lower) and HotMoE-$\mathbf{W}_g$ (higher), respectively. As shown in Table 3, all variants suffer performance drops and yield negative MRD values, confirming the importance of both routing mechanisms and the overall advantage of our hybrid routing schema.

Second, we analyze the effect of different components in the auxiliary loss. The results in Table 4 show that the synergistic use of $\mathcal{L}_{\text{sim}}$ alongside GJS yields better overall performance in AVG and MRD. This suggests that combining these losses not only maintains expert load balance and selection certainty, but also guides the router to learn more precise task-expert matching patterns. Appendix E.1 presents additional results demonstrating similar trends of the two ablation experiments in task synergy scenarios.

4.5 Enhancing Other Methods with Similarity-Guided Loss

As shown in Table 5, incorporating $\mathcal{L}_{\text{sim}}$ consistently improves performance when applied to both the standard $\mathbf{W}_g$ router and MoLORA. This demonstrates that $\mathcal{L}_{\text{sim}}$ is a generalizable enhancement that benefits different methods by encouraging more semantically coherent expert activations.

5 Conclusion

We proposed HotMoE, a hybrid MoE-LoRA framework for multi-task instruction tuning. It mitigates task interference through a hybrid routing scheme and a similarity-guided auxiliary loss. Specifically, in lower layers, expert-expert attention routing was designed to capture cross-task collaborative relations among LoRA experts. While in higher layers, token-expert attention routing was presented to align fine-grained task semantics with specialized experts. HotMoE delivers substantial gains in accuracy and obtains positive MRD on two multi-task instruction tuning scenarios covering seven NLP benchmarks.

Acknowledgments

The research was supported by Natural Science Foundation of China (62272403).

References

Aoki, R.; Tung, F.; and Oliveira, G. L. 2022. Heterogeneous multi-task learning with expert diversity. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 19(6): 3093–3102.

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Chen, Z.; Shen, Y.; Ding, M.; Chen, Z.; Zhao, H.; Learned-Miller, E. G.; and Gan, C. 2023. Mod-Squad: Designing Mixtures of Experts As Modular Multi-Task Learners. In *CVPR*, 11828–11837.

Clark, C.; Lee, K.; Chang, M.-W.; Kwiatkowski, T.; Collins, M.; and Toutanova, K. 2019. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. In *NAACL*, 2924–2936.

Clark, P.; Cowhey, I.; Etzioni, O.; Khot, T.; Sabharwal, A.; Schoenick, C.; and Tafjord, O. 2018. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *CoRR*, abs/1803.05457.

Dagan, I.; Roth, D.; Zanzotto, F.; and Sammons, M. 2022. *Recognizing textual entailment: Models and applications*. Springer Nature.

Dolan, B.; and Brockett, C. 2005. Automatically constructing a corpus of sentential paraphrases. In *International Workshop on Paraphrasing*.

Fedus, W.; Zoph, B.; and Shazeer, N. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120): 1–39.

Feng, W.; Hao, C.; Zhang, Y.; Han, Y.; and Wang, H. 2024. Mixture-of-LoRAs: An efficient multitask tuning method for large language models. In *COLING*, 11371–11380.

Gao, C.; Chen, K.; Rao, J.; Sun, B.; Liu, R.; Peng, D.; Zhang, Y.; Guo, X.; Yang, J.; and Subrahmanian, V. 2024. Higher layers need more lora experts. *arXiv preprint arXiv:2402.08562*.

Gou, Y.; Liu, Z.; Chen, K.; Hong, L.; Xu, H.; Li, A.; Yeung, D.-Y.; Kwok, J. T.; and Zhang, Y. 2023. Mixture of cluster-conditional lora experts for vision-language instruction tuning. *arXiv preprint arXiv:2312.12379*.

Gupta, S.; Mukherjee, S.; Subudhi, K.; Gonzalez, E.; Jose, D.; Awadallah, A. H.; and Gao, J. 2022. Sparsely activated mixture-of-experts are robust multi-task learners. *arXiv preprint arXiv:2204.07689*.

Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; De Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; and Gelly, S. 2019. Parameter-efficient transfer learning for NLP. In *ICML*, 2790–2799.

Hu, E. J.; yelong shen; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR*.

Jacobs, R. A.; Jordan, M. I.; Nowlan, S. J.; and Hinton, G. E. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1): 79–87.

Junczys-Dowmunt, M. 2019. Microsoft translator at WMT 2019: Towards large-scale document-level neural machine translation. *arXiv preprint arXiv:1907.06170*.

Lepikhin, D.; Lee, H.; Xu, Y.; Chen, D.; Firat, O.; Huang, Y.; Krikun, M.; Shazeer, N.; and Chen, Z. 2021. {GS}hard: Scaling Giant Models with Conditional Computation and Automatic Sharding. In *ICLR*.

Liao, M.; Chen, W.; Shen, J.; Guo, S.; and Wan, H. 2025. HMoRA: Making LLMs More Effective with Hierarchical Mixture of LoRA Experts. In *ICLR*.

Lin, B. Y.; Zhou, W.; Shen, M.; Zhou, P.; Bhagavatula, C.; Choi, Y.; and Ren, X. 2020. CommonGen: A Constrained Text Generation Challenge for Generative Commonsense Reasoning. In *EMNLP*, 1823–1840.

Lin, C.-Y.; and Och, F. J. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *ACL*, 605–612.

Lin, J. 1991. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information theory*, 37(1): 145–151.

Ling, C.; Zhao, X.; Lu, J.; Deng, C.; Zheng, C.; Wang, J.; Chowdhury, T.; Li, Y.; Cui, H.; Zhang, X.; et al. 2023. Domain specialization as the key to make large language models disruptive: A comprehensive survey. *arXiv preprint arXiv:2305.18703*.

Liu, Q.; Wu, X.; Zhao, X.; Zhu, Y.; Xu, D.; Tian, F.; and Zheng, Y. 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*, 1104–1114.

Liu, Z.; and Luo, J. 2024. Adamole: Fine-tuning large language models with adaptive mixture of low-rank adaptation experts. *arXiv preprint arXiv:2405.00361*.

Luo, T.; Lei, J.; Lei, F.; Liu, W.; He, S.; Zhao, J.; and Liu, K. 2024. Moelora: Contrastive learning guided mixture of experts on parameter-efficient fine-tuning for large language models. *arXiv preprint arXiv:2402.12851*.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *ACL*, 311–318.

Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; and Dean, J. 2017. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *ICLR*.

Shen, S.; Hou, L.; Zhou, Y.; Du, N.; Longpre, S.; Wei, J.; Chung, H. W.; Zoph, B.; Fedus, W.; Chen, X.; Vu, T.; Wu, Y.; Chen, W.; Webson, A.; Li, Y.; Zhao, V. Y.; Yu, H.;

Keutzer, K.; Darrell, T.; and Zhou, D. 2024. Mixture-of-Experts Meets Instruction Tuning: A Winning Combination for Large Language Models. In *ICLR*.

Tang, A.; Shen, L.; Luo, Y.; Yin, N.; Zhang, L.; and Tao, D. 2024. Merging Multi-Task Models via Weight-Ensembling Mixture of Experts. *International Conference on Machine Learning*.

Taori, R.; Gulrajani, I.; Zhang, T.; Dubois, Y.; Li, X.; Guestrin, C.; Liang, P.; and Hashimoto, T. B. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*.

Tian, C.; Shi, Z.; Guo, Z.; Li, L.; and Xu, C.-Z. 2024. Hydralora: An asymmetric lora architecture for efficient fine-tuning. *Neural Information Processing Systems*, 37: 9565–9584.

Wang, X.; Zhao, H.; Wang, S.; Wang, H.; and Liu, Z. 2024. MALoRA: Mixture of Asymmetric Low-Rank Adaptation for Enhanced Multi-Task Learning. *arXiv preprint arXiv:2410.22782*.

Wei, J.; Bosma, M.; Zhao, V.; Guu, K.; Yu, A. W.; Lester, B.; Du, N.; Dai, A. M.; and Le, Q. V. 2022. Finetuned Language Models are Zero-Shot Learners. In *ICLR*.

Wu, S.; Luo, J.; Chen, X.; Li, L.; Zhao, X.; Yu, T.; Wang, C.; Wang, Y.; Wang, F.; Qiao, W.; et al. 2024. Yuan 2.0-m32: Mixture of experts with attention router. *arXiv preprint arXiv:2405.17976*.

Wu, X.; Huang, S.; and Wei, F. 2024. Mixture of LoRA Experts. In *ICLR*.

Xu, J.; Lai, J.; and Huang, Y. 2025. MeteoRA: Multiple-tasks Embedded LoRA for Large Language Models. In *ICLR*.

Xu, L.; Xie, H.; Qin, S.-Z. J.; Tao, X.; and Wang, F. L. 2023. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*.

Yang, A.; Yang, B.; Hui, B.; Zheng, B.; Yu, B.; Zhou, C.; Li, C.; Li, C.; Liu, D.; Huang, F.; et al. 2024. Qwen2 Technical Report. *arXiv preprint arXiv:2407.10671*.

Ye, Q.; Zha, J.; and Ren, X. 2022. Eliciting and understanding cross-task skills with task-level mixture-of-experts. *arXiv preprint arXiv:2205.12701*.

Yu, T.; Kumar, S.; Gupta, A.; Levine, S.; Hausman, K.; and Finn, C. 2020. Gradient surgery for multi-task learning. In *NeurIPS*, 5824–5836.

Zadouri, T.; Üstün, A.; Ahmadian, A.; Ermis, B.; Locatelli, A.; and Hooker, S. 2024. Pushing Mixture of Experts to the Limit: Extremely Parameter Efficient MoE for Instruction Tuning. In *ICLR*.

Zellers, R.; Holtzman, A.; Bisk, Y.; Farhadi, A.; and Choi, Y. 2019. HellaSwag: Can a Machine Really Finish Your Sentence? In *ACL*, 4791–4800.

Zhao, L.; Zeng, W.; Shi, X.; and Zhou, H. 2024. MoSLD: An Extremely Parameter-Efficient Mixture-of-Shared LoRAs for Multi-Task Learning. *arXiv preprint arXiv:2412.08946*.

Zhao, Z.; Zhou, Y.; Zhu, D.; Shen, T.; Wang, X.; Su, J.; Kuang, K.; Wei, Z.; Wu, F.; and Cheng, Y. 2025. Each Rank Could be an Expert: Single-Ranked Mixture of Experts LoRA for Multi-Task Learning. *arXiv preprint arXiv:2501.15103*.

Zhao, Z.; Ziser, Y.; and Cohen, S. B. 2024. Layer by layer: Uncovering where multi-task learning happens in instruction-tuned large language models. *arXiv preprint arXiv:2410.20008*.