From Outcome to Process: Optimizing MoE Load Balancing with MCTS

Wenjun Ke[1,2], **Hengyuan Xu**[3*], **Ziyu Shang**[1], **Yao He**[4],
Jiahao Wang[5], **Zijie Xu**[1], **Peng Wang**[1,2], **Yuhang Lou**[1], **Jiajun Liu**[1]
[1]School of Computer Science and Engineering, Southeast University, China
[2]Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications (Southeast University), Ministry of Education, China
[3]College of Software Engineering, Southeast University, China
[4]Institute of Collaborative Innovation, University of Macau, Macau, China
[5]Baidu
{kewenjun, xuhengyuan, ziyus1999, wang_jh, zijiexu, pwang, 220242470, jiajliu}@seu.edu.cn

Abstract

Mixture of Experts (MoE) dynamically routes inputs to specialized expert networks, enabling large language models to scale capacity with low inference overhead. To further improve MoE's parameter efficiency in resource-constrained scenarios, LoRA–MoE integrates LoRA for lightweight adaptation while preserving MoE's specialization. Despite these benefits, the effectiveness of LoRA–MoE still hinges on balanced expert utilization, where certain experts dominate activations while most remain underutilized. Existing balancing strategies focus on constraining the final distribution of expert usage, but overlook the routing decisions made at each layer. As a result, imbalances gradually accumulate across the routing hierarchy. To address this challenge, we propose LayerMoE, a novel three-stage framework that leverages *process-level* rewards to guide balanced expert routing. Specifically, to overcome the limitation of focusing only on final losses and ignoring intermediate routing, we introduce Monte Carlo Tree Search (MCTS)-based sampling that decomposes *outcome-level* supervision into layer-wise reward signals, guiding expert choices throughout the routing process. For efficiency, we organize Transformer layers into groups, which constrain the search space of MCTS and keep exploration overhead tractable while retaining the hierarchical structure. Extensive experiments on representative datasets (e.g., ARC, RACE, OBQA) show that applying LayerMoE consistently improves the performance of state-of-the-art LoRA-MoE baselines, yielding an average accuracy gain of 1.39%. Notably, the maximum improvement reaches 2.50%.

1 Introduction

Background. Mixture of Experts (MoE) (Lepikhin et al., 2020; Zhu et al., 2024; Li et al., 2025a; Ning et al., 2025; Zhang et al., 2025) is a scalable and parameter-efficient architecture for large language models (LLMs). In MoE architecture, only a subset of experts is activated for each input, which reduces inference cost compared with dense models. Recent studies combine MoE with low-rank adaptation (LoRA) (Hu et al., 2022) to achieve parameter-efficient training of LLMs, forming LoRA-MoE models (Tian et al., 2024; Dou et al., 2024; Wu et al., 2024) that leverage MoE's modular specialization and LoRA's efficient task adaptation. Despite these benefits, the effectiveness of LoRA–MoE models depends on balanced expert utilization, which remains a challenge: certain experts are frequently selected whereas others are rarely activated, reducing parameter efficiency. We empirically examine this imbalance in LoRA–MoE models by analyzing expert activations across four domains: mathematics, astronomy, medicine, and commonsense reasoning (Figure 1). In all domains, expert activations follow a similar pattern, with few experts repeatedly chosen and the majority rarely

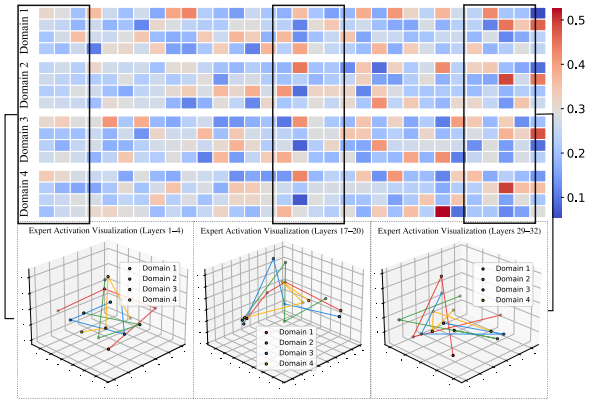

Figure 1: Empirical analysis of LoRA–MoE expert activations reveals a load-balancing issue, where expert activation remains similar across different inputs, particularly in deeper layers, indicating limited expert specialization. The bottom (left to right) shows activations for Layers 1–4, 17–20, and 29–32. (see Appendix C).

*Corresponding authors.

Findings of the Association for Computational Linguistics: ACL 2026, pages 28831–28848

engaged. This collapse of expert usage indicates that current training fails to maintain balanced expert utilization, thereby limiting the intended capacity gains of LoRA–MoE.

Motivation. To address the load-balancing issue, existing approaches introduce localized balancing constraints (Dou et al., 2024) or distribution matching (Bai et al., 2024) in LoRA–MoE. These methods implement an implicit load-balancing constraint operating at the *outcome level* (Cobbe et al., 2021b; Hosseini et al., 2024; Setlur et al., 2024). These approaches can be understood by analogy to reinforcement learning (RL), where MoE routing resembles a sequential decision process: each router makes a layer-wise choice of experts. When balancing is enforced only at the *outcome level*, errors occur in individual routing steps, such as the repeated activation of certain experts. Similar to the *Outcome Reward Model* (ORM) in RL, balancing is optimized solely at the *outcome level*. By contrast, *Process Reward Model* (PRM) provides feedback at each decision step, suggesting the need for layer-wise balancing to prevent progressive accumulation of imbalance across layers. In light of the transition from ORM to PRM, we propose a layer-wise balancing mechanism, *LayerMoE*, that shifts focus from *outcome-level* optimization to a *process-level*. By formulating expert selection as a sequential decision process, rewards are assigned at each routing step. With such process-oriented perspective, *LayerMoE* facilitates balanced expert utilization throughout the routing hierarchy.

Our Approach. We propose *LayerMoE*, a three-stage framework for PRM-guided LoRA–MoE training that integrates Monte Carlo Tree Search (MCTS) (Li et al., 2025b; Zhao et al., 2023; Wu et al., 2025) to guide expert routing decisions. First, the Transformer layers are partitioned into non-overlapping groups, and experts within each group are aggregated into macro-experts. This grouping reduces the search complexity of MCTS and improves efficiency by restricting the action space during routing. Second, expert selection is formulated as a sequential decision-making process. MCTS explores the routing space by iteratively selecting experts based on the upper confidence bound (UCB) criterion and expanding unexplored nodes. It then simulates full routing paths and back-propagates the corresponding rewards. The back-propagated rewards are used during training to update the parameters governing path-selection strategies. Finally, we use MCTS-derived scores as *process-level* reward signals under the PRM formulation. These signals are combined with router outputs to form the final gating scores, enabling efficient and interpretable expert allocation.

Our Contributions. This work makes three main contributions:

- We identify and characterize the *expert collapse* phenomenon in LoRA–MoE models, where a small subset of experts dominates activation while others remain unused.
- We propose a novel MCTS-based hierarchical guidance mechanism that shifts from ORM-style *outcome-level* balancing to PRM-style *process-level* balancing, introducing a sampling phase that enforces load balancing at each layer without additional training cost.
- We evaluate LayerMoE on 8 datasets across 5 domains, showing it outperforms baselines with better expert diversity and accuracy.

2 Method

We present a three-stage framework for reward-guided training of Mixture-of-Experts (MoE) models augmented with LoRA and MCTS. Our objective is to learn an LoRA-MoE Transformer f_θ whose routing decisions combine router outputs with MCTS-derived expert scores, while achieving efficiency through a group-wise routing strategy.

2.1 Preliminaries

Task Definition. Let $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{N}$ denote the training data, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. We consider a k-layer Transformer $\mathcal{T} = \{\tau^1, \ldots, \tau^k\}$, where each layer τ^l contains an LoRA-MoE submodule with E experts $A_1^l, \ldots, A_E^l$. Each expert is parameterized as a low-rank adaptation applied to the shared feed-forward transformation.

$$\tau^l(x_i) = F^l(x_i) + \sum_{j=1}^{E} g_j^l(x_i) \cdot A_j^l(x_i) \quad (1)$$

where $g^l(x_i) \in \Delta^{E-1}$ is the gating distribution.

Formally, the training objective is

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{train}}} \left[\mathcal{L}(f_\theta(x), y)\right] \quad (2)$$

where f_θ denotes the LoRA-MoE Transformer parameterized by θ, and $\mathcal{L}$ is a supervised loss.

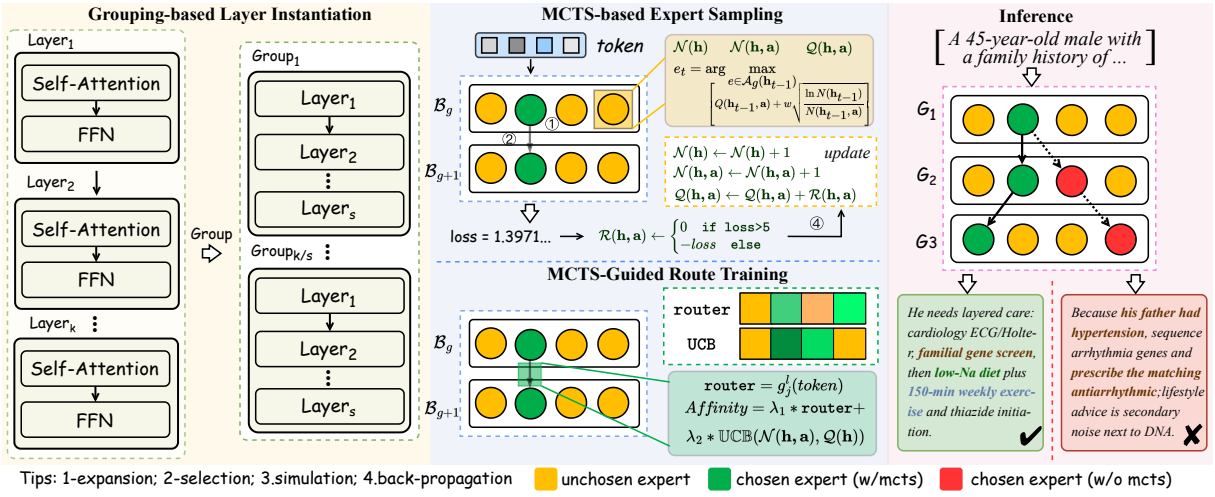

Figure 2: Overview of LayerMoE framework. The arrows indicate the direction of data flow.

MCTS. MCTS incrementally constructs a search tree through four steps: `selection`, `expansion`, `simulation` and `back-propagation`. A decision path π explored by MCTS can be represented as

$$\pi = \{\mathbf{a}^1, \ldots, \mathbf{a}^k\}, \quad \mathbf{a}^l \in \mathcal{A}^l \tag{3}$$

where $\mathbf{a}^l$ denotes the action selected at step l, and $\mathcal{A}^l$ is the candidate action set at that step. In the search tree, each node $\mathbf{h}$ corresponds to a state in the decision process. For each state–action pair, MCTS maintains statistics including: $\mathcal{Q}(\mathbf{h}, \mathbf{a})$, $\mathcal{N}(\mathbf{h}), \mathcal{N}(\mathbf{h}, \mathbf{a}), \mathcal{R}(\mathbf{h}, \mathbf{a})$ representing action-value estimate, the visit count of node $\mathbf{h}$, the visit count of edge $(\mathbf{h}, \mathbf{a})$, and cumulative reward, respectively. Each state $\mathbf{h}$ can be represented by a triplet $\rho = \{\mathcal{A}(\mathbf{h}), \mathcal{N}(\mathbf{h}), \mathcal{Q}(\mathbf{h}, \mathcal{A}^l)\}$. In our framework, to shift from *outcome-level* to *process-level* modeling in MoE routing, we employ MCTS to formulate expert selection as a sequential decision process.

2.2 Grouping-based Layer Instantiation

LoRA-MoE parameterization. Each expert in the LoRA–MoE is implemented as a LoRA adapter on top of a shared base weight. Let $W^l \in \mathbb{R}^{d\times d}$ denote the backbone weight at layer l. The j-th expert is parameterized via low-rank adaptation

$$\begin{aligned} &\Delta W_j^l = \alpha B_j^l A^l, \\ &A^l \in \mathbb{R}^{r\times d},\ B_j^l \in \mathbb{R}^{d\times r},\ r \ll d \end{aligned} \tag{4}$$

where A^l is shared across experts within the same layer and B_j^l is expert-specific. The effective weight applied to input x_i under routing distribution $g = (g_1, \ldots, g_E)$ is

$$W_{\text{eff}}^l = W^l + \sum_{j=1}^{E} g_j \Delta W_j^l \tag{5}$$

Here, the expert corresponds to the LoRA adapter ΔW_j^l, with the backbone weight W^l unchanged.

Layer grouping. We partition the full set of k layers into a sequence of non-overlapping groups, each treated as a decision step $\mathbf{a}^l$ in the routing process, which reduces the complexity of the Monte Carlo search tree and lowers the computational overhead.

$$\begin{aligned} &\mathcal{S} = \mathcal{B}_0, \ldots, \mathcal{B}_{g-1}, \\ &\mathcal{B}_g = \{\tau^{gs}, \ldots, \tau^{(g+1)s-1}\} \end{aligned} \tag{6}$$

where $\mathcal{S}$ denotes the set of layer groups, $\mathcal{B}_g$ represents the g-th group containing s consecutive layers indexed from τ^{gs} to $\tau^{(g+1)s-1}$, and $G = \lceil k/s \rceil$ denotes the total number of groups. Each group $\mathcal{B}_g$ is associated with a set of macro-experts $\mathcal{E}_g = \{\mathcal{E}_{g,1}, \ldots, \mathcal{E}_{g,E}\}$, where E denotes the number of macro-experts per group. This grouping reduces the complexity of the routing search space by enabling shared routing decisions within group.

Output of Stage #1. The process produces a grouped mixture-of-experts backbone f_θ augmented with LoRA parameters, where each group $\mathcal{B}_g$ is associated with a corresponding macro-expert set $\mathcal{E}_g$. This grouped structure defines the action space $\mathcal{A}$ for the subsequent MCTS, enabling efficient exploration of expert paths across groups.

2.3 MCTS-based Expert Sampling

We formulate expert routing as a sequential decision-making problem, transforming it from an *outcome-level* to a *process-level* representation, and employ MCTS to guide the exploration of the routing space π.

Selection. To apply MCTS to the MoE routing process, we treat each decision step $\mathbf{a}$ as choosing a expert for a group of consecutive layers. At step t, the next expert $\mathbf{a}_t$ is selected based on the Upper Confidence Bound (UCB) criterion:

$$\mathbf{a}_t = \arg \max_{\mathbf{a} \in \mathcal{A}_g(\mathbf{h}_{t-1})} \left[\mathcal{Q}(\mathbf{h}_{t-1}, \mathbf{a}) + w \sqrt{\frac{\ln \mathcal{N}(\mathbf{h}_{t-1})}{\mathcal{N}(\mathbf{h}_{t-1}, \mathbf{a})}} \right] \tag{7}$$

where $\mathcal{A}_g(\mathbf{h}_{t-1})$ denotes the set of candidate experts for group $\mathcal{B}_g$ given partial state $\mathbf{h}_{t-1}$, and $w > 0$ is a hyperparameter controlling the balance between exploration and exploitation.

Expansion. Expansion grows the routing decision path π by sampling candidate experts $\mathcal{E}_g$ at each step:

$$\{e_t^{(b)}\}_{b=1}^{E} \subset \mathcal{E}_g \tag{8}$$

where b indexes the sampled candidate experts for the current expansion step. These sampled experts define the new child nodes to be added to the search tree, incrementally building the routing path π.

Simulation. A complete routing path π_i is generated by a rollout procedure, which sequentially samples candidate expert $\mathcal{E}_g$ for each group $\mathcal{B}_g$. This path defines a full expert routing for the input x_i and produces a corresponding reward $\mathcal{R}_i$:

$$\begin{aligned} \mathit{loss}_i &= \mathcal{L}(f_\theta(x_i; \pi_i), y_i), \\ \mathcal{R}_i &= \begin{cases} 0, & \text{if } \mathit{loss}_i > \gamma \\ -\mathit{loss}_i, & \text{otherwise} \end{cases} \end{aligned} \tag{9}$$

where γ is a threshold introduced to prevent excessive losses at the early stages of training from dominating the reward signal, thereby mitigating cold-start instability.

Back-propagation. The reward $\mathcal{R}_i$ from a rollout is propagated backward along the routing path π_i, updating the corresponding states in the search tree:

$$\begin{aligned} \mathcal{Q}(\mathbf{h}_t, \mathbf{a}) &\leftarrow \mathcal{Q}(\mathbf{h}_{t-1}, \mathbf{a}) + \delta \mathcal{R}_i, \\ \mathcal{N}(\mathbf{h}_t) &\leftarrow \mathcal{N}(\mathbf{h}_{t-1}) + 1, \\ \mathcal{N}(\mathbf{h}_t, \mathbf{a}) &\leftarrow \mathcal{N}(\mathbf{h}_{t-1}, \mathbf{a}) + 1 \end{aligned} \tag{10}$$

where $\delta \in (0, 1)$ is a smoothing factor controlling the update rate of $\mathcal{Q}(\mathbf{h}_t, \mathbf{a})$.

proposition 1 (Efficiency of MCTS-Guided Routing)**.** *Let Π be the routing space and suppose each node h admits actions $\mathcal{A}(h)$ with expected rewards $\{\mu_{h,a}\}_{a \in \mathcal{A}(h)}$. For each non-optimal action $a \neq a_h^*$, define the gap $\Delta_{h,a} = \mu_{h,a_h^*} - \mu_{h,a} > 0$. Then the cumulative regret of UCT selection at node h after T visits satisfies*

$$R_h(T) \leq \sum_{a \neq a_h^*} \frac{C \ln T}{\Delta_{h,a}} \tag{11}$$

which is sublinear in T. In contrast, greedy or random policies can incur $\Omega(T)$ regret.

Proof Sketch. The UCT rule is equivalent to UCB1 at each node, inheriting its logarithmic regret bound, while greedy lacks exploration and random lacks exploitation.

Output of Stage #2. The process produces a set of accumulated triplet statistics $\rho = \{\mathcal{A}(\mathbf{h}), \mathcal{N}(\mathbf{h}), \mathcal{Q}(\mathbf{h}, \mathcal{A}^l)\}$ for each macro-expert $\mathcal{E}_g$. These statistics are used for optimizing expert routing in subsequent stages.

2.4 MCTS-Guided LoRA-MoE Training

We incorporate the statistics collected from `Stage #2` into the gating mechanism of the LoRA-MoE model, formulating expert selection as a *process-level* decision problem. This integration applies *process-level* modeling to MoE routing via MCTS-driven sampling, enabling reward-informed expert selection that balances learned gating with exploration.

Reward-regularized gating. The gating score for expert E_j^l at layer l for input x_i is defined as a convex combination of the standard gating output $g_j^l(x_i)$ and the MCTS-derived score:

$$\tilde{g}_j^l(x_i) = \lambda g_j^l(x_i) + (1 - \lambda)\, \mathbb{UCB}_j^l \tag{12}$$

where $\lambda \in [0, 1]$ controls the interpolation ratio. Here, $\mathbb{UCB}$ denotes the exploration–exploitation score defined in Eq.7. This reward-regularized gating allows the routing policy to adapt dynamically by combining the learned gating function with MCTS search statistics.

Final objective. The training objective integrates the modified gating scores into the standard supervised learning loss:

$$\begin{aligned} &\mathcal{J}(\theta, \{\mathbf{A}^e\}, \{\mathbf{B}^e\}) = \\ &\mathbb{E}_{(x,y) \sim \mathcal{D}_{\text{train}}} \left[\mathcal{L}(f_{\theta, \{\mathbf{A}^e\}, \{\mathbf{B}^e\}}(x; \tilde{g}_j^l(x)), y) \right] \end{aligned} \tag{13}$$

where $f_{\theta,\{\mathbf{L}^e\},\{\mathbf{R}^e\}}(\cdot)$ denotes the LoRA-MoE model parameterized by the base model parameters θ and the expert-specific LoRA factors $\{\mathbf{A}^e\}$ and $\{\mathbf{B}^e\}$.

Output of Stage #3. Stage #3 uses MCTS to incorporate *process-level* modeling into MoE routing, producing a trained LoRA-MoE model $f_{\theta,\{\mathbf{A}^e\},\{\mathbf{B}^e\}}$ with reward-regularized, interpretable gating and efficient specialization across layer groups $\{\mathcal{B}_g\}$.

3 Experiment

3.1 Datasets and Settings

Datasets. We evaluate our model on a diverse set of tasks spanning five key domains. For language understanding and reasoning, we use the ARC-Challenge and ARC-Easy datasets (Clark et al., 2018). For reading comprehension, the model is trained and evaluated on RACE-high and RACE-middle (Lai et al., 2017). For code generation, we use HumanEval (Chen et al., 2021). For closed-book question answering, we consider OBQA (Mihaylov et al., 2018) and BoolQ (Clark et al., 2019). For mathematics, we use GSM8K (Cobbe et al., 2021a). During fine-tuning, the hyperparameter settings are: lora rank = 8, lora heads = 4, training epochs = 3, batch size = 8, learning rate = 1e-5, and dropout rate = 0.05. All experiments are conducted on 1 × L20 (48GB) GPUs. We adopt LLaMA3.2-1B and LLaMA3-8B (Dubey et al., 2024) as the base models.

Baseline. We apply our LayerMoE framework to the following MoE-based SFT methods for LLMs. **HydraLoRA** (Tian et al., 2024) adopts an asymmetric LoRA structure that shares a global low-rank matrix while learning task-specific components, effectively balancing parameter sharing and specialization. **LoRAMoE** (Dou et al., 2024) augments frozen backbones with multiple LoRA experts and a routing network, mitigating catastrophic forgetting during continual fine-tuning. **MoSLoRA** (Wu et al., 2024) incorporates sparse gating with LoRA, updating only a small subset of experts to achieve parameter-efficient adaptation across tasks. **MixLoRA** (Li et al., 2024) combines LoRA experts through mixture strategies, enabling adaptive expert selection and improved transfer across heterogeneous datasets. **MoLA** (Gao et al., 2025) introduces modular LoRA adapters that can be dynamically composed, enhancing flexibility in multi-domain fine-tuning. **GMoE** (Bai et al., 2024) leverages global gating signals to coordinate expert activation across layers, improving efficiency and consistency in large-scale MoE fine-tuning.

3.2 Main Results

In our main experiments, we conduct MoE-based SFT on LLaMA3.2-1B and LLaMA3-8B. We present the results on NLU, NLG, QA tasks and STEM-related domains (Code, Math) in Table 1.

Firstly, incorporating our MCTS-based inter-layer optimization yields consistent improvements across both small and large backbones. On LLaMA3.2-1B, HydraLoRA improves its average score from 36.96% to 39.38% (2.42%), while LoRAMoE increases from 37.69% to 40.19% (2.50%). At the 8B scale, HydraLoRA gains 2.33% (66.23% to 68.56%) and GMoE 1.38% (67.55% to 68.93%). These representative cases illustrate that the benefit of MCTS holds across both weak and strong baselines.

Secondly, the magnitude of improvement varies with model capacity and task type. Weaker backbones tend to benefit more: for instance, HydraLoRA-1B shows 6.71% gain on HumanEval pass@10 (27.44% to 34.15%), whereas HydraLoRA-8B achieves a smaller but still notable 3.42% gain (35.36% to 38.78%). A similar pattern is observed in math reasoning, where GSM8K improves by 3.03% on 1B (26.46% to 29.49%) versus 2.20% on 8B (48.82% to 51.02%). This trend suggests that MCTS-guided path exploration is especially helpful when the backbone or expert routing is relatively weak.

Thirdly, our framework generalizes across domains, with particularly strong effects on reasoning-heavy benchmarks. On RACE-m, MoSLoRA-1B improves substantially from 57.87% to 63.02% (5.18%), while HydraLoRA-8B improves from 59.47% to 67.83% (8.36%). In contrast, on easier classification-style datasets such as BoolQ, the gains are modest (e.g., HydraLoRA-8B improves only 0.36%). This contrast highlights that our approach contributes most in tasks requiring multi-step reasoning rather than shallow classification.

3.3 Analysis Experiments

Effect under Different Sampling Size. To examine the impact of the sampling size in our MCTS-based MoE routing, we analyze performance under varying batch sizes from 2 to 32. As shown in Figure 3, the baseline accuracy of 8B model increases

Method	NLU		NLG		Code	QA		Math	Avg.
	ARC-e	ARC-c	RACE-m	RACE-h	HumanEval	OBQA	BoolQ	GSM8K	
LLaMA3.2									
HydraLoRA-1B	55.01	22.35	49.44	30.16	13.78/27.44	23.60	61.25	26.46	36.96
w/ **MCTS**	54.42	26.02(3.67↑)	51.95(2.51↑)	29.87	13.29/34.15(6.71↑)	28.60(5.0↑)	60.52	29.49(3.03↑)	39.38(2.42↑)
LoRAMoE-1B	56.64	22.35	41.92	30.16	14.51/31.10	22.20	65.41	31.77	37.69
w/ **MCTS**	56.44	24.91↑(2.56↑)	44.36(2.44↑)	31.76(1.60↑)	13.23/36.58(5.48↑)	26.20(4.0↑)	66.42(1.01↑)	34.87(3.10↑)	40.19(2.50↑)
MoSLoRA-1B	54.63	26.11	57.87	31.19	17.13/31.10	25.20	60.03	26.16	39.04
w/ **MCTS**	56.02(1.4↑)	25.85	63.02(5.18↑)	32.48(1.29↑)	18.78(1.65↑) /32.31(1.21↑)	27.60(2.4↑)	60.52(0.49↑)	28.13(1.97↑)	40.74(1.70↑)
MixLoRA-1B	56.73	29.09	61.56	43.37	12.74/21.95	49.80	63.88	28.51	44.36
w/ **MCTS**	61.32(4.59↑)	35.41(6.32↑)	61.84(0.28↑)	43.34	14.63(1.89↑) /23.78(1.83↑)	50.60(0.8↑)	63.30	29.80(1.29↑)	46.17(1.81↑)
MoLA-1B	62.03	33.02	59.81	44.40	12.93/24.39	50.20	63.42	26.61	45.49
w/ **MCTS**	62.88(0.85↑)	35.01(1.99↑)	60.19(0.38↑)	44.77(0.37↑)	13.54(0.61↑) /26.22(1.83↑)	50.80(0.60↑)	63.73(0.31↑)	25.54	46.14(0.65↑)
GMoE-1B	60.44	29.27	57.80	42.02	13.41/21.34	48.60	58.44	26.38	43.04
w/ **MCTS**	61.66(1.22↑)	31.91(2.64↑)	58.64(0.84↑)	41.80	14.63/22.56(1.22↑)	47.40	61.80(3.36↑)	27.75(1.37↑)	44.19(1.15↑)
LLaMA3									
HydraLoRA-8B	72.94	71.59	59.47	55.23	35.36/70.12	78.20	73.43	48.82	66.23
w/ **MCTS**	75.55(2.61↑)	69.80	67.83(8.36↑)	61.35(6.12↑)	38.78(3.42↑) /72.56(2.44↑)	76.60	73.79(0.36↑)	51.02(2.20↑)	68.56(2.33↑)
LoRAMoE-8B	76.09	74.49	76.25	61.84	34.76/68.90	78.20	73.55	51.02	70.04
w/ **MCTS**	78.87(2.78↑)	76.62(2.13↑)	80.15(3.90↑)	57.09	31.10/70.73(1.83↑)	79.60(1.40↑)	74.13(0.58↑)	50.49	70.96(0.92↑)
MoSLoRA-8B	74.83(3.49↑)	70.31	80.78	52.83	34.15/71.34	77.60	70.73	53.90	69.04
w/ **MCTS**	78.32	72.10(1.79↑)	81.69(0.91↑)	54.66(1.83↑)	37.20(3.05↑) /73.78(2.44↑)	75.80	69.85	54.20(0.30↑)	70.05(1.01↑)
MixLoRA-8B	85.65	75.68	72.49	58.60	36.59/72.56	81.40	69.85	52.77	71.13
w/ **MCTS**	83.25	77.13(1.45↑)	73.54(1.05↑)	56.38	32.95/76.22(3.66↑)	80.00	70.55(0.70↑)	52.99(0.22↑)	71.26(0.13↑)
MoLA-8B	83.41	76.19	78.69	61.26	24.39/65.24	82.40	71.04	45.26	70.44
w/ **MCTS**	84.34(0.93↑)	75.09	77.92	59.87	23.78/70.73(5.49↑)	83.00(0.6↑)	70.73	47.08(1.82↑)	71.10(0.66↑)
GMoE-8B	81.06	76.96	66.85	58.55	32.32/62.80	80.20	68.59	45.41	67.55
w/ **MCTS**	82.11(1.05↑)	78.41(1.45↑)	68.94(2.09↑)	56.52	29.88/66.46(3.66↑)	82.20(2.0↑)	68.84(0.25↑)	47.99(2.58↑)	68.93(1.38↑)

Table 1: Comparison of performance on 8 datasets in NLU, NLG, Code, QA and Math tasks (accuracy, %). Bold and underline indicate the best and second-best scores, respectively. More results see Appendix D and E

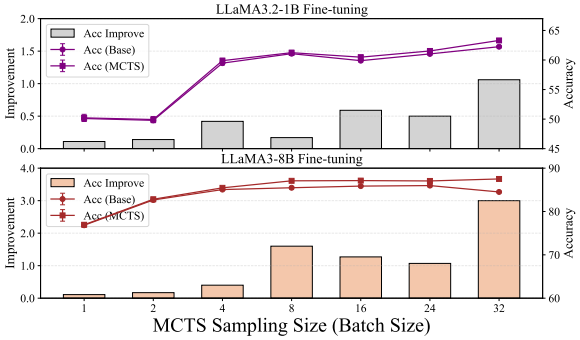

Figure 3: Effect of different sampling sizes on model performance. Bars (left y-axis) show the accuracy improvement (*Inc*) of MCTS over the baseline; lines (right y-axis) show accuracies (*Acc*) for MCTS and baseline.

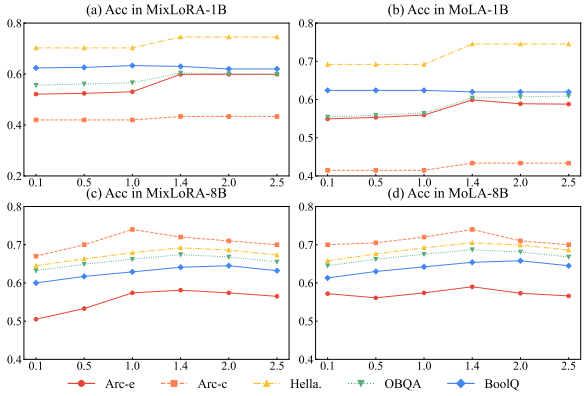

Figure 4: Effect of different exploration parameters on model performance. The y-axis represents model accuracy under varying exploration settings.

from 76.83% at batch size 1 to 85.86% at batch size 16, after which it plateaus around 85%. Our MCTS-based sampling method similarly peaks at batch size 16 with 87.13%, but consistently outperforms the baseline across all batch sizes. The relative improvement grows with batch size, ranging from +0.11% at batch size 1 to +1.27% at batch size 16, suggesting that larger batches provide more informative probability distributions for MCTS exploration and enable more effective expert routing while maintaining robustness across configurations.

Effect under Different Exploration Parameter. To examine the impact of the exploration parameter in our MCTS-based MoE routing, we conduct experiments using the LLaMA3-8B model and LLaMA3.2-1B on several QA datasets. We vary the exploration coefficient w in the UCB formula from 0.1 to 2.5 and measure model performance in terms of accuracy. As shown in Figure 4, the accuracy improve as w increases up to 1.4, indicating that moderate exploration helps the agent find more effective expert paths. However, overly large values of w can lead to suboptimal routing due to excessive exploration. This demonstrates that tuning the exploration parameter is crucial for balancing exploitation and exploration in our MCTS-based expert selection.

Effect under Different Group Size. To assess the influence of the group size s in the layer grouping scheme, we vary s from 2 to 16. Table 2 shows the accuracy of the LLaMA3-8B model with different group configurations. Smaller groups increase the action space for MCTS and can potentially im-

Size	RACE-h		OBQA		BoolQ	
	Time	Acc	Time	Acc	Time	Acc
2	3:13:41	42.62	0:20:17	**50.60**	0:24:56	63.17
4	2:53:33	**43.34**	0:19:33	50.20	0:23:17	**63.30**
8	2:37:16	39.42	0:19:31	49.40	0:22:56	62.13
16	2:32:20	38.62	0:18:57	48.20	0:22:13	60.91

Table 2: Effect of different group sizes on model performance. Time (in hours:minutes:seconds) indicates fine-tuning duration; Acc denotes accuracy (%).

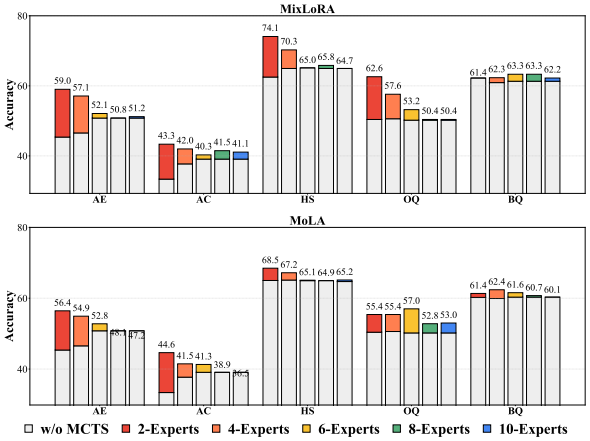

Figure 5: Effect of different expert number on model accuracy across datasets. AE: ARC-Easy, AC: ARC-Challenge, HS: HellaSwag, OQ: OBQA, BQ: BoolQ.

prove routing flexibility with higher computational cost, while larger groups reduce complexity but may limit routing granularity. Our experiments indicate that a group size of 4 provides a good trade-off between efficiency and performance.

Effect under Different Number of Experts. Finally, we study how the number of experts E per MoE layer affects performance. We train models with E ranging from 2 to 10 and evaluate on multiple datasets. Figure 5 shows that increasing the number of experts generally improves accuracy, as it provides richer representations and more flexibility in expert routing. However, the marginal gain diminishes beyond a certain point, suggesting that an appropriate number of experts balances model capacity and efficiency. These results confirm that our MCTS-guided MoE routing effectively leverages multiple experts to improve model performance.

4 Case Study

To further validate the effectiveness of LayerMoE with MCTS, we conduct a case study covering two representative reasoning settings: (1) **cross-domain reasoning**, which requires integrating knowledge from heterogeneous fields, and (2) **multi-step reasoning**, which involves sequential sub-decisions and intermediate verification. We analyze both the static parameter specialization and the dynamic routing behaviors, providing quantitative insights into how MCTS improves expert coordination and interpretability.

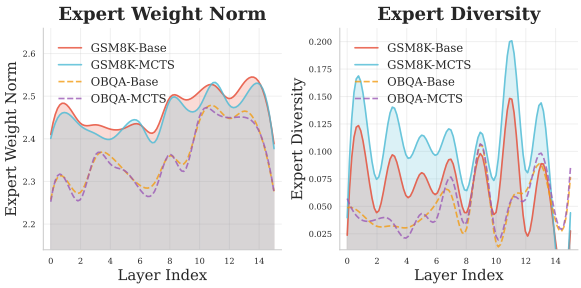

Figure 6: Static parameter analysis in a case study with OBQA corresponding to cross-domain and GSM8K corresponding to multi-step, where (a) expert weight norm, (b) expert diversity, and (c) expert balance are shown across layers (Details see Appendix F).

Static Parameter Analysis. We first examine the learned LoRA parameters and expert activation patterns after training, with computation details provided in the Appendix B. For the **cross-domain** task (OBQA), as shown in Figure 6. the baseline LoRA-MoE tends to concentrate activations within a small subset of experts, leading to redundant specialization. With MCTS-guided routing, expert usage becomes more balanced, improving both diversity and load balance: expert diversity increases from 0.0506 to 0.0531 (+4.94%), and balance from 0.0215 to 0.0227 (+5.58%), while the overall expert weight norm remains stable (–0.3%). This indicates that MCTS introduces a mild regularization effect, encouraging more uniform activation without disrupting learned representations.

For the **multi-step reasoning** task (GSM8K), the difference is more pronounced. MCTS substantially enhances expert diversity from 0.0666 to 0.0988 (+48.35%) and balance from 0.0270 to 0.0402 (+48.89%), while the expert weight norm slightly decreases (–0.6%), suggesting a more efficient specialization. These results show that MCTS exerts stronger influence on reasoning-oriented experts, promoting clearer functional separation across layers-early layers focus on retrieval and parsing, while later layers activate computation-oriented experts.

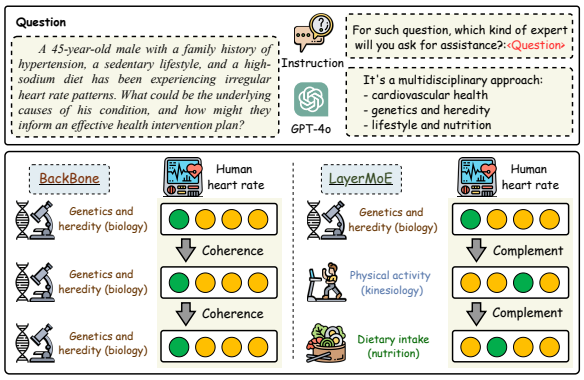

Figure 7: A cross-domain case in biology, kinesiology, and nutrition. Baseline shows static expert activation while LayerMoE promotes diverse expert engagement.

Dynamic Inference Analysis. We conduct a case study on **cross-domain reasoning** to illustrate how MCTS-guided routing dynamically reallocates experts during inference. For the clinical query, the baseline shows uniform activation across layers, repeatedly relying on a narrow subset of experts regardless of contextual variation. In contrast, MCTS-guided LayerMoE exhibits higher activation counts and greater diversity across layers, indicating that more experts are effectively involved in the reasoning process.

This case illustrates that MCTS-guided LayerMoE better captures the multi-disciplinary nature of the query. As shown in Figure 7, while the baseline model repeatedly invokes experts from a single domain (e.g., genetics and heredity), MCTS encourages broader participation across complementary domains such as physical activity and nutrition. This diversified activation pattern reflects more balanced knowledge integration and aligns with human reasoning behavior, where solving complex cross-domain problems typically involves combining insights from multiple expert perspectives. Layer-wise activation heatmaps further confirm that MCTS smooths the activation distribution and alleviates expert over-concentration, leading to more balanced expert utilization (Figure 8).

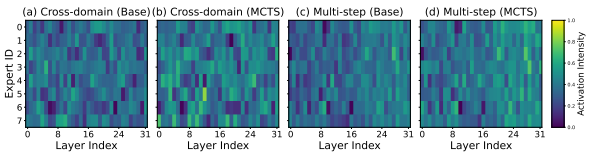

Figure 8: Case study of expert activation: LayerMoE enhances activation diversity in cross-domain (+2.17%) and multi-step (+1.33%) reasoning.

5 Related Work

5.1 Mixture of Experts

Mixture of Experts (MoE) (He et al., 2025; Kong et al., 2024; Jiang et al., 2024; Seo et al., 2025; Yu et al., 2025; Liu et al., 2025) grows large language models (LLMs) by selectively activating subnetworks to increase compute efficiency without inference costs. Knowledge hybridity and knowledge redundancy, where experts' expertise dilutes specialization and underused parameters hinder learning, limit MoE designs. Recent studies propose structural and training changes to address these issues (Lu et al., 2024; Chowdhury et al., 2024; Xie et al., 2024). DeepSeekMoE (Dai et al., 2024) splits FFNs into more, smaller experts and applies shared expert isolation and modularity, improving performance compared to dense models. LLaMA-MoE (Zhu et al., 2024) avoids full-scale retraining by modularly splitting and pretraining dense models like LLaMA-2 into MoE structures, conserving prior knowledge and fostering expert specialisation. MoE-LLaVA (Lin et al., 2024) dynamically assigns visual and textual tokens to modality-specific experts using soft routing to vision-language models (VLMs). Our work contributes to addressing knowledge hybridity by introducing an explicit Topic that enhances expert specificity.

5.2 Monte Carlo Tree Search.

Monte Carlo Tree Search (MCTS) (Coulom, 2006; Kocsis and Szepesvári, 2006; Browne et al., 2012) balances exploration and exploitation by building search trees through randomized simulations, enabling effective planning in large state spaces. However, issues such as scalability to high-branching domains and inefficiency of rollouts limit its broader adoption. Recent studies propose algorithmic augmentations to address these challenges (Yao et al., 2023; Li et al., 2025c; Hu et al., 2025). For example, Tree-of-Thoughts (ToT) (Yao et al., 2023) introduces structured look ahead and backtracking for LLM reasoning; MCTS-Process (Li et al., 2025c) leverages process supervision with MCTS to improve reasoning transfer; and MCTS-RAG (Hu et al., 2025) enhances retrieval-augmented generation by guiding document selection through search. In the vision domain, SOTA with Less (Wang et al., 2025) applies MCTS-guided sample selection to achieve data-efficient visual reasoning self-improvement. Our work contributes to this line by leveraging MCTS

as a sampling mechanism to enhance expert routing in mixture-structured language model training.

6 Conclusion

This paper proposes *LayerMoE*, a training framework that integrates MCTS with LoRA-MoE models through *process-level* rewards derived from MCTS exploration to enable reward-regularized and interpretable expert routing. This method encourages balanced expert utilization and mitigates load-balancing during training, leading to more diverse and adaptive expert behaviors. Extensive experiments across 8 datasets demonstrate that LayerMoE consistently outperforms strong LoRA-MoE baselines, achieving better accuracy, higher expert diversity, and improved interpretability.

Limitation

Our experiments are conducted on models up to 8B parameters due to limited computational resources. While this scale is sufficient to demonstrate the effectiveness of MCTS-guided routing and LayerMoE's interpretability, we acknowledge that larger foundation models (e.g., LLaMA-70B or Mixtral-12x7B) could further reveal the scalability and robustness of the proposed framework. Extending to denser or hierarchical MoE architectures (e.g., multi-group or task-aware routing) would require substantially more GPU memory and longer optimization cycles, which were beyond our available resources.

Acknowledgement

We would like to thank the anonymous reviewers for their insightful comments. This work was supported by National Science Foundation of China (Grant Nos.62376057), the Start-up Research Fund of Southeast University (RF1028623234) and the Big Data Computing Center of Southeast University.

References

Ting Bai, Yue Yu, Le Huang, Zenan Xu, Zhe Zhao, and Chuan Shi. 2024. Gmoe: Empowering llms fine-tuning via moe graph collaboration. *arXiv preprint arXiv:2412.16216*.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, and Henrique Ponde de Oliveira Pinto. 2021. Evaluating large language models trained on code.

Mohammed Nowaz Rabbani Chowdhury, Meng Wang, Kaoutar El Maghraoui, and Wang Naigang. 2024. A provably effective method for pruning experts in fine-tuned sparse mixture-of-experts. In *ICML*.

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 *NAACL*.

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:1803.05457v1*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021a. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021b. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Rémi Coulom. 2006. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer.

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

DeepSeek-AI. 2024. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Shihan Dou, Enyu Zhou, Yan Liu, and Gao Songyang. 2024. LoRAMoE: Alleviating world knowledge forgetting in large language models via MoE-style plugin. In *ACL*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Ruibo Liu, and Baochen Sun. 2025. MoLA: MoE LoRA with layer-wise expert allocation. In *Findings of the*

Association for Computational Linguistics: NAACL 2025.

Yifei He, Yang Liu, Chen Liang, and Hany Hassan Awadalla. 2025. Efficiently editing mixture-of-experts models with compressed experts. *arXiv preprint arXiv:2503.00634*.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. 2024. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*.

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.

Yunhai Hu, Yilun Zhao, Chen Zhao, and Arman Cohan. 2025. Mcts-rag: Enhancing retrieval-augmented generation with monte carlo tree search. *arXiv preprint arXiv:2503.20757*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, and Mensch Arthur. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer.

Rui Kong, Yuanchun Li, Qingtian Feng, and Wang Weijun. 2024. SwapMoE: Serving off-the-shelf MoE-based large language models with tunable memory budget. In *ACL*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, and Chen Dehao. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.

Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan Zhang, Lei Duan, Jie Zuo, Cal Yang, and Mingjie Tang. 2024. Mixlora: Enhancing large language models fine-tuning with lora-based mixture of experts. *Preprint*, arXiv:2404.15159.

Dengchun Li, Naizheng Wang, Zihao Zhang, and Haoyang Yin. 2025a. Dynmole: Boosting mixture of lora experts fine-tuning with a hybrid routing mechanism. *arXiv preprint arXiv:2504.00661*.

Peiji Li, Kai Lv, Yunfan Shao, Yichuan Ma, Linyang Li, Xiaoqing Zheng, Xipeng Qiu, and Qipeng Guo. 2025b. Fastmcts: A simple sampling strategy for data synthesis. *arXiv preprint arXiv:2502.11476*.

Shuangtao Li, Shuaihao Dong, Kexin Luan, Xinhan Di, and Chaofan Ding. 2025c. Enhancing reasoning through process supervision with monte carlo tree search. *arXiv preprint arXiv:2501.01478*.

Bin Lin, Zhenyu Tang, Yang Ye, and Cui Jiaxi. 2024. Moe-llava: Mixture of experts for large vision-language models. *arXiv preprint arXiv:2401.15947*.

Zehua Liu, Han Wu, Ruifeng She, and Xiaojin Fu. 2025. Molae: Mixture of latent experts for parameter-efficient language models. *arXiv preprint arXiv:2503.23100*.

Xudong Lu, Qi Liu, Yuhui Xu, and Zhou Aojun. 2024. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. In *ACL*.

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 *EMNLP*.

Lin Ning, Harsh Lara, Meiqi Guo, and Abhinav Rastogi. 2025. MoDE: Effective multi-task parameter efficient fine-tuning with a mixture of dyadic experts. In *NAACL*.

Jean Seo, Jaeyoon Kim, and Hyopil Shin. 2025. MoFE: Mixture of frozen experts architecture. In *NAACAL*.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2024. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*.

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

Xiyao Wang, Zhengyuan Yang, Chao Feng, Hongjin Lu, Linjie Li, Chung-Ching Lin, Kevin Lin, Furong Huang, and Lijuan Wang. 2025. Sota with less: Mcts-guided sample selection for data-efficient visual reasoning self-improvement. *arXiv preprint arXiv:2504.07934*.

Fang Wu, Weihao Xuan, Heli Qi, Ximing Lu, Aaron Tu, Li Erran Li, and Yejin ChoiRetry. 2025. Deepsearch: Overcome the bottleneck of reinforcement learning with verifiable rewards via monte carlo tree search. *arXiv preprint arXiv:2509.25454*.

Taiqiang Wu, Jiahao Wang, Zhe Zhao, and Ngai Wong. 2024. Mixture-of-subspaces in low-rank adaptation. In *EMNLP*.

Zhitian Xie, Yinger Zhang, Chenyi Zhuang, and Shi Qitao. 2024. Mode: A mixture-of-experts model with mutual distillation among the experts. In *AAAI*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Beiming Yu, Zhenfei Yang, and Xiushuang Yi. 2025. MoKA:parameter efficiency fine-tuning via mixture of kronecker product adaption. In *COLING*.

Junyi Zhang, Chuanhu Ma, Xiong Wang, Yuntao Nie, Yuqing Li, Yuedong Xu, Xiaofei Liao, Bo Li, and Hai Jin. 2025. Popfetcher: Towards accelerated mixture-of-experts training via popularity based expert-wise prefetch. In *2025 USENIX Annual Technical Conference (USENIX ATC 25)*, pages 1053–1069.

Zirui Zhao, Wee Sun Lee, and David Hsu. 2023. Large language models as commonsense knowledge for large-scale task planning. *Advances in neural information processing systems*, 36:31967–31987.

Tong Zhu, Xiaoye Qu, Daize Dong, and Ruan Jiacheng. 2024. Llama-moe: Building mixture-of-experts from llama with continual pre-training. In *EMNLP*.

Appendix Contents

A Datasets

Overview. We evaluate LAYERMOE on eight datasets spanning five representative domains, covering reasoning, comprehension, and generation abilities. Table 3 summarizes the dataset statistics, task types, and evaluation objectives. All datasets are publicly available and widely adopted in prior literature. We follow the standard training–validation–testing splits provided by the original benchmarks.

Domains and Tasks.

- **Language understanding & reasoning:** ARC-Challenge and ARC-Easy (Clark et al., 2018) test scientific and commonsense reasoning under multiple-choice settings.
- **Reading comprehension:** RACE-middle and RACE-high (Lai et al., 2017) assess long-context understanding through passage–question–answer triples.
- **Closed-book QA:** OBQA (Mihaylov et al., 2018) and BoolQ (Clark et al., 2019) require factual and Boolean reasoning without context retrieval.
- **Code generation:** HumanEval (Chen et al., 2021) measures functional correctness of Python code generation tasks.
- **Mathematical reasoning:** GSM8K (Cobbe et al., 2021a) evaluates multi-step arithmetic reasoning over grade-school math problems.

Remarks. The chosen datasets collectively cover both *cross-domain* (e.g., OBQA) and *multi-step reasoning* (e.g., GSM8K) scenarios, which are crucial for evaluating expert specialization and routing dynamics in LAYERMOE. This diversity ensures that the observed improvements in expert diversity and balance are not confined to a specific task type or modality.

B Calculation Metrics

To analyze the effects of MCTS-guided routing on expert utilization, we perform a post-hoc static analysis of the learned LoRA–MoE parameters after training. Three complementary metrics are computed: **Expert Weight Norm**, **Expert Diversity**, and **Expert Balance**. All statistics are averaged across 16 layers and 4 experts per layer (each containing three LoRA components: `gate_proj`, `up_proj`, and `down_proj`).

B.1 Expert Weight Norm

The *Expert Weight Norm* quantifies the magnitude or strength of each expert's learned parameters. For an expert e at layer l, let its LoRA weight matrices be (A_e^l, B_e^l). The effective expert weight is $\Delta W_e^l = B_e^l A_e^l$, and its norm is defined as:

$$\|\Delta W_e^l\|_2 = \sqrt{\sum_{i,j} (\Delta W_{e,ij}^l)^2}. \tag{14}$$

We compute the mean value across all experts and layers:

$$\text{WeightNorm} = \frac{1}{LE} \sum_{l=1}^{L} \sum_{e=1}^{E} \|\Delta W_e^l\|_2. \tag{15}$$

Interpretation. A higher weight norm indicates stronger expert activation and larger influence on the output, while a lower norm suggests weaker contribution or implicit regularization. Consistent with our observations in Figure 6, MCTS-guided routing produces slightly smaller weight norms (–0.3% on average), implying a compact yet effective parameter configuration. This reduction acts as an implicit regularization effect, discouraging over-amplified weights and promoting stability across layers.

B.2 Expert Diversity

Expert Diversity measures the dissimilarity among experts' effective parameters, capturing the extent to which each expert learns distinct transformations. For layer l, the diversity is computed as:

$$\text{Diversity}(l) = \frac{2}{E(E-1)} \sum_{e_1 < e_2} \left(1 - \frac{\langle \Delta W_{e_1}^l, \Delta W_{e_2}^l \rangle}{\|\Delta W_{e_1}^l\|_2 \, \|\Delta W_{e_2}^l\|_2}\right), \tag{16}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product between flattened weight matrices. The overall expert diversity is averaged across layers:

$$\text{Diversity} = \frac{1}{L} \sum_{l=1}^{L} \text{Diversity}(l). \tag{17}$$

Interpretation. Higher diversity indicates that experts specialize on distinct subspaces, leading to better coverage of heterogeneous patterns. As shown in our cross-domain analysis (OBQA task), MCTS-guided routing improves expert diversity from 0.0506 to 0.0531 (+4.94%), suggesting that MCTS promotes more varied expert behaviors without destabilizing training.

Dataset	Domain	Task Type	Size (Train / Test)	Answer Format
ARC-Easy (Clark et al., 2018)	Reasoning	Multiple-choice QA	2.3K / 2.5K	4 options
ARC-Challenge (Clark et al., 2018)	Reasoning	Multiple-choice QA	1.1K / 1.1K	4 options
RACE-middle (Lai et al., 2017)	Reading comprehension	Passage QA	25K / 1.4K	4 options
RACE-high (Lai et al., 2017)	Reading comprehension	Passage QA	62K / 3.5K	4 options
OBQA (Mihaylov et al., 2018)	Closed-book QA	Science reasoning	5.0K / 500	4 options
BoolQ (Clark et al., 2019)	Closed-book QA	Boolean QA	9.4K / 3.2K	Yes/No
HumanEval (Chen et al., 2021)	Code generation	Code synthesis	– / 164	Python function
GSM8K (Cobbe et al., 2021a)	Math reasoning	Open-ended QA	7.5K / 1.3K	Numeric text

Table 3: Summary of datasets used in our experiments.

B.3 Summary of Observations

Together, these metrics provide a static diagnostic of how MCTS-guided routing alters the LoRA–MoE parameter landscape:

- **Expert Weight Norm**: slightly reduced, indicating mild implicit regularization and more compact parameter scaling.
- **Expert Diversity**: increased, reflecting greater functional differentiation among experts.
- **Expert Balance**: improved, revealing better load sharing across experts.

Overall, the results demonstrate that MCTS-guided routing enhances both diversity and balance with negligible cost to weight magnitude, confirming its effectiveness as a process-level regularizer for LoRA–MoE models.

C Expert Activation Analysis for LoRA–MoE

To investigate expert utilization patterns in LoRA–MoE, we analyze the activation distributions of the **LLaMA-3-8B** model fine-tuned with the LoRA–MoE configuration. The analysis covers four representative domains—mathematics, astronomy, medicine, and commonsense reasoning—corresponding to the input cases listed in Table 4.

C.1 Experimental Setup

We employ the LLaMA-3-8B model fine-tuned under the LoRA–MoE setup on mixed-domain instruction data. Each input query is passed through the model, and we record the router activation probabilities (i.e., gating scores) for all experts at each MoE layer. Token-level activations are averaged within each layer, yielding a per-layer expert activation profile per domain case.

To visualize domain-specific activation patterns, expert activation vectors from selected layers are normalized and then projected into a two-dimensional latent space through a *nonlinear manifold-preserving transformation*. Specifically:

- **Normalization:** Each expert activation vector is normalized across experts to highlight relative routing preferences rather than absolute magnitudes.
- **Projection:** The normalized vectors are embedded into a 2D space using a manifold-preserving method (t-SNE), capturing similarity among expert activations across different domains.

The resulting visualization (Figure 1) reveals that LoRA–MoE experts exhibit domain-dependent clustering, indicating partial specialization yet noticeable overlap among domains—suggesting that expert routing remains imbalanced across tasks.

C.2 Domain Cases

These domain-specific cases in Table 4 are used to analyze expert activations in LoRA–MoE (LLaMA-3-8B).

D Qwen as backbone model

For backbone diversity, we added experiments on Qwen3-0.6B and Qwen3-4B.

As shown in Table 5, LayerMoE consistently improves performance over the Base HydraLoRA across all datasets and backbones, with gains ranging from +0.89% to +3.68%. Smaller models (Qwen3-0.6B) see larger relative improvements, while larger models (Qwen3-4B) still benefit meaningfully. These results demonstrate architecture-agnostic effectiveness, indicating that LayerMoE captures general routing dynamics rather than model-specific quirks.

Domain	Input Case
Mathematics	Solve the differential equation: $\frac{d^2y}{dx^2} + 4\frac{dy}{dx} + 5y = 0$, given initial conditions $y(0) = 1$, $y'(0) = 0$.
Astronomy	**Problem.** Solve the second-order differential equation: $\frac{d^2\theta}{dt^2} + \frac{GM}{r(t)^3}\theta = 0$, given initial conditions $\theta(0) = \theta_0$, $\theta'(0) = 0$, where $r(t)$ is the distance function of a small orbiting object with perihelion distance $r_0 = 3.5 \times 10^{10}$ m and eccentricity $e = 0.9$.
Medicine	A 45-year-old male presents with stage 1 hypertension (BP 142/92 mmHg). Recommend first-line pharmacological treatment according to ACC/AHA guidelines.
Commonsense Reasoning	A person places an ice cube in a glass of water and leaves it at room temperature. Describe what happens after one hour and explain why.

Table 4: Domain-specific cases used for analyzing expert activations in LoRA–MoE (LLaMA-3-8B).

Dataset	HydraLoRA	/w MCTS	Increase (%)
Qwen3-0.6B			
ARC-E	19.65	23.33	+3.68
RACE-m	27.03	29.19	+2.16
OpenBookQA	25.40	28.20	+2.80
Qwen3-4B			
ARC-E	53.53	54.42	+0.89
RACE-m	58.31	60.11	+1.80
OpenBookQA	56.60	59.20	+2.60

Table 5: Supplementary Backbone Results

E Appendix Baselines

To further validate our approach, we have included the following additional experiments in Table 6: Random expert reassignment and Auxiliary-Loss-Free Load Balancing (DeepSeek-AI, 2024)

LayerMoE still achieves the largest gain, consistently improving accuracy across all three datasets—particularly on OBQA, where it outperforms Baseline by up to 5 points—showing that a structured, search-based approach is more effective than random expert reassignment and other simple load-balancing heuristics.

F Diversity Analysis

To further analyze the wave-like oscillations observed in Figure 6, we computed expert diversity

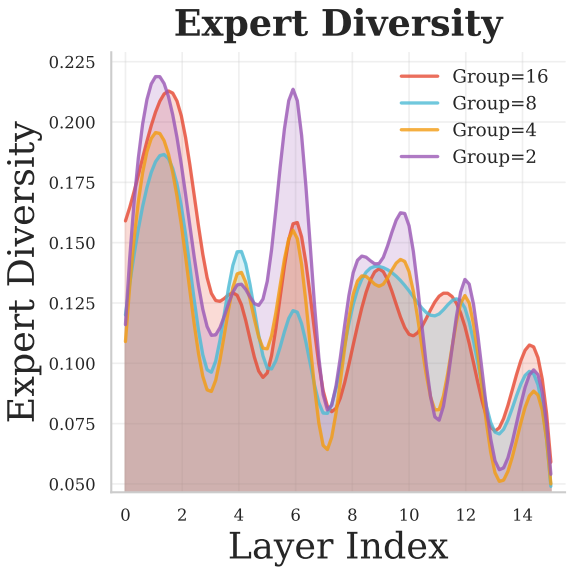

Figure 9: Expert diversity under different group size

curves for different group sizes.

From Figure 9, we observe that smaller group sizes lead to higher variance across layers. This means that the mild "wave-like" oscillations become more pronounced as the group size decreases. This trend aligns with the wave-like fluctuations and further confirms that the oscillations arise from inherent gating dynamics rather than from the shared-routing design itself.

The following table presents the diversity comparison between the base model and our method across various tasks.

As shown in Table 7, reasoning-intensive tasks like ARC-C/E achieve the largest diversity gains ($\sim$ 95%), while knowledge-style reading-comprehension tasks (RACE-m/h) show a more modest increase ($\sim$ 20%/ $\sim$ 11%). This pattern aligns with the figure 6 that GSM8K benefits more than OBQA: tasks demanding multi-step reasoning tend to activate a broader set of experts under MCTS exploration, whereas fact-retrieval-style questions rely on fewer, highly specialised experts, limiting the observable diversity delta. Importantly, even modest improvements (e.g., OBQA) correlate with better routing stability and accuracy as shown in Table 7 and Figure 6, indicating that increased diversity whether large or small has a consistent positive effect across tasks.

G Training Details

Table 8 summarizes the overall training statistics of LayerMoE compared to its Base counterpart across five representative reasoning and understand-

Method	ARC-c		RACE-m		OBQA	
	Base	increase	Base	increase	Base	increase
llama3.2-1B						
Base MoE (HydraLoRA)	22.35	–	49.44	–	23.60	–
Random expert reassignment	21.95	-0.40	47.31	-2.13	19.80	-3.80
Auxiliary-Loss-Free Load Balancing	25.09	+2.74	52.36	+2.92	21.40	-2.20
LayerMoE (ours)	26.02	+3.67	51.95	+2.51	28.60	+5.00
Base MoE (LoRAMoE)	22.35	–	41.92	–	22.20	–
Random expert reassignment	17.12	-5.23	37.74	-4.18	18.60	-3.60
Auxiliary-Loss-Free Load Balancing	23.51	+1.16	42.21	+0.29	22.40	+0.20
LayerMoE (ours)	24.91	+2.56	44.36	+2.44	26.20	+4.00

Table 6: Supplementary Baseline Results (Accuracy, %) Across Three Datasets (llama3-1B)

Task	Base	Ours	Increase
BoolQA	0.044259	0.065875	+48.84%
Code	0.079422	0.090531	+13.99%
RACE-m	0.057550	0.069471	+20.71%
RACE-h	0.072821	0.081032	+11.28%
ARC-c	0.021728	0.042371	+95.01%
ARC-e	0.022574	0.044685	+97.95%

Table 7: Task Diversity Comparison (Base vs. Ours)

ing benchmarks. For each task, we report the total floating-point operations (GFLOPs), final training loss, and wall-clock training time. Across both the LLaMA-3.2-1B and LLaMA-3-8B models, LayerMoE introduces a slight increase in computational overhead due to expert routing and MCTS-based gating, yet maintains a comparable training loss to the Base model.

To provide an explicit cost breakdown between rollouts, reward estimation, and finetuning, we use HydraLoRA w/ MCTS (llama3-8B as backbone) fine-tuning in the ARC-E dataset as a case.

As shown in Table 9, MCTS rollouts and reward estimation together account for only 6.12% of the total fine-tuning time, while fine-tuning dominates with 92.67%. This demonstrates that the additional overhead from MCTS is minimal, and compute fairness relative to the baseline is largely preserved.

H Algorithm

The proposed LAYERMOE framework is trained in three stages, as summarized in Algorithm 1. In Stage #1, we initialize LoRA experts and group transformer layers to form macro-expert structures. Stage #2 employs Monte Carlo Tree Search (MCTS) to explore expert combinations across groups, collecting reward-regularized routing statistics that quantify each expert's contribution and uncertainty. Finally, Stage #3 integrates these MCTS-derived scores with learned router gates to form reward-guided gating signals, enabling more balanced and interpretable expert specialization during fine-tuning. This design allows MCTS exploration to guide LoRA–MoE optimization with minimal overhead.

Algorithm 1 Three-Stage Reward-Guided LoRA–MoE Training with MCTS

Require: Training data $\mathcal{D}$, layers $\{\tau^1, \ldots, \tau^k\}$, experts E, group size s, MCTS budget B, UCB weight w, interpolation λ, reward threshold γ, smoothing δ, optimizer $\mathcal{O}$

Ensure: Trained model $f_{\theta,\{\mathbf{A}^e,\mathbf{B}^e\}}$

Stage 1: Grouping & Initialization
Partition layers into groups $\mathcal{S} = \{\mathcal{B}_g\}_{g=1}^{G}$, $G = \lceil k/s \rceil$
for each layer l **do**
    Initialize frozen weights W^l, shared LoRA A^l, and expert-specific B_j^l $(j = 1..E)$
end for
Stage 2: MCTS-based Expert Exploration
for each batch $\mathcal{B} \subset \mathcal{D}$ **do**
    for each input $x_i \in \mathcal{B}$ (parallelizable) **do**
        Initialize root node h_0
        for $t = 1$ to B **do**
            Selection: $a_t \leftarrow \arg\max_a [\mathcal{Q}(h,a) + w\sqrt{\ln \mathcal{N}(h)/\mathcal{N}(h,a)}]$
            Expansion: add child, sample remaining groups $\Rightarrow$ path π_i
            Rollout: compute loss $\mathcal{L}(f_\theta(x_i;\pi_i), y_i)$ and reward $\mathcal{R}_i = \mathbf{1}_{loss_i<\gamma}(-loss_i)$
            Backprop: update $\mathcal{Q}, \mathcal{N}$ with smoothing δ
        end for
        Store statistics $\rho_i = \{\mathcal{A}, \mathcal{N}, \mathcal{Q}\}$
    end for
end for
Stage 3: Reward-Guided LoRA–MoE Training
for each batch $\mathcal{B}$ **do**
    for each $x_i \in \mathcal{B}$ **do**
        Get gating $g_j^l(x_i)$ and MCTS score $\mathbb{UCB}_j^l$ from ρ_i
        Fuse: $\tilde{g}_j^l(x_i) \leftarrow \lambda g_j^l(x_i) + (1-\lambda)\mathbb{UCB}_j^l$
    end for
    Compute loss $\mathcal{J} = \frac{1}{|\mathcal{B}|}\sum \mathcal{L}(f_\theta(x_i;\tilde{g}), y_i)$
    Update $\theta, \{A^l, B_j^l\}$ via optimizer $\mathcal{O}$
end for
return $f_{\theta,\{\mathbf{A}^e,\mathbf{B}^e\}}$

Task	Model	Variant	Train Loss	Runtime
GSM8K	LLaMA-3.2-1B	LayerMoE	0.7594	00:39:05
	LLaMA-3.2-1B	Base	0.7386	00:36:31
	LLaMA-3-8B	LayerMoE	0.5512	04:56:50
	LLaMA-3-8B	Base	0.5369	04:41:39
ARC-Challenge	LLaMA-3.2-1B	LayerMoE	1.6337	00:21:07
	LLaMA-3.2-1B	Base	1.5810	00:19:46
	LLaMA-3-8B	LayerMoE	1.2067	04:05:23
	LLaMA-3-8B	Base	1.2495	03:57:29
ARC-Easy	LLaMA-3.2-1B	LayerMoE	2.4645	00:41:51
	LLaMA-3.2-1B	Base	2.3616	00:39:37
	LLaMA-3-8B	LayerMoE	1.9370	06:04:40
	LLaMA-3-8B	Base	1.7170	05:47:02
RACE-Middle	LLaMA-3.2-1B	LayerMoE	0.3233	03:07:25
	LLaMA-3.2-1B	Base	0.3182	03:12:32
	LLaMA-3-8B	LayerMoE	0.3217	12:39:16
	LLaMA-3-8B	Base	0.3195	12:12:30
RACE-High	LLaMA-3.2-1B	LayerMoE	0.2029	04:06:42
	LLaMA-3.2-1B	Base	0.1892	03:56:35
OBQA	LLaMA-3.2-1B	LayerMoE	0.3622	00:59:27
	LLaMA-3.2-1B	Base	0.3512	00:51:50
	LLaMA-3-8B	LayerMoE	0.2188	12:14:22
	LLaMA-3-8B	Base	0.2280	11:55:43

Table 8: Training statistics (LayerMoE vs Base).

Component	**Time (%)**	**Notes**
MCTS rollout	4.23%	Pure forward pass, frozen parameters
Reward estimation	1.89%	Lightweight scoring function
finetuning	92.67%	Dominant cost
Other overhead	1.2%	Data loading, synchronization
Total	100%	–

Table 9: Time Breakdown of Components with Notes