GuiLoMo: Allocating Expert Number and Rank for LoRA-MoE via Bilevel Optimization with GuidedSelection Vectors

Xinrong Chen[1]*, **Hengyuan Zhang**[2]*, **Yingmin Qiu**[3], **Xiao Liang**[4], **Ziyue Li**[5], **Guanyu Wang**[1], **Weiping Li**[1], **Tong Mo**[1]†, **Hayden Kwok-Hay So**[2], **Ngai Wong**[2]†
[1]Peking University, [2]The University of Hong Kong,
[3]Beijing University of Posts and Telecommunications, [4]University of California, Los Angeles,
[5]National Science Library, Chinese Academy of Sciences
chenxinrong23@stu.pku.edu.cn hengyuan.zhang88@gmail.com

Abstract

Parameter-efficient fine-tuning (PEFT) methods, particularly Low-Rank Adaptation (LoRA), offer an efficient way to adapt large language models with reduced computational costs. However, their performance is limited by the small number of trainable parameters. Recent work combines LoRA with the Mixture-of-Experts (MoE), i.e., LoRA-MoE, to enhance capacity, but two limitations remain in hindering the full exploitation of its potential: 1) the influence of downstream tasks when assigning expert numbers, and 2) the uniform rank assignment across all LoRA experts, which restricts representational diversity. To mitigate these gaps, we propose GuiLoMo, a fine-grained layer-wise expert numbers and ranks allocation strategy with GuidedSelection Vectors (GSVs). GSVs are learned via a prior bilevel optimization process to capture both model- and task-specific needs, and are then used to allocate optimal expert numbers and ranks. Experiments on three backbone models across diverse benchmarks show that GuiLoMo consistently achieves superior or comparable performance to all baselines. Further analysis offers key insights into how expert numbers and ranks vary across layers and tasks, highlighting the benefits of adaptive expert configuration. Our code is available at https://github.com/Liar406/Gui-LoMo.git.

1 Introduction

Although large language models (LLMs) have demonstrated remarkable performance across a wide range of general tasks (Jiang et al., 2023; Chowdhery et al., 2023; Jian et al., 2023; Touvron et al., 2023b; Xiong et al., 2023), they still fall short in certain tasks or domains, such as reasoning (Tong et al., 2024; Srivastava and Gandhi, 2024; Yu et al., 2025; Li et al., 2025; Liang et al., 2025), multilingualism (Huang et al., 2023; Gurgurov et al., 2024; Zhang et al., 2024a), and text generation in specialized contexts (Biancotti et al., 2024; Zhang et al., 2024b; Yang et al., 2024a,b; Li et al., 2024b,a; Wang et al., 2024; Chang et al., 2025). To enhance the performance of LLMs in these challenging areas, a common practice is fine-tuning. However, with the growing size of current LLMs, full fine-tuning faces significant challenges in terms of computational efficiency and memory consumption. To mitigate these issues, parameter-efficient fine-tuning (PEFT) methods have gained considerable attention (Houlsby et al., 2019; Li and Liang, 2021; Lester et al., 2021; Hu et al., 2022; Liu et al., 2022; Zhang et al., 2023; Yang et al., 2025). Among these methods, Low-Rank Adaptation (LoRA) (Hu et al., 2022) is regarded as one of the most efficient approaches. Nonetheless, its performance remains constrained due to the relatively small number of trainable parameters (Xu et al., 2023). Recent studies suggest that combining LoRA with the Mixture-of-Experts (MoE) paradigm, referred to as LoRA-MoE, by incorporating multiple LoRA modules, offers a promising solution to this limitation (Wu et al., 2024; Gao et al., 2024; Qing et al., 2024; Dou et al., 2024; Liu et al., 2023; Luo et al., 2024).

	MoLA	AlphaLoRA	GuiLoMo (Ours)
Model Specific	✗	✓	✓
Task Specific	✗	✗	✓
Expert Number	✓	✓	✓
Expert Rank	✗	✗	✓

Table 1: Compared to existing methods, our proposed GuiLoMo strategy can allocate the optimal expert numbers and ranks within LoRA-MoE, tailored to specific models and tasks.

However, fully exploiting the potential of LoRA-MoE remains an open research question. First, Gao

* Equal contribution. Order set by random dice roll.
† Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 7552–7567
November 4-9, 2025

et al. (2024) considered that uniformly allocating the number of experts across all layers is suboptimal, as different layers play distinct roles in the model. Over-allocating experts to certain layers can lead to redundancy and degraded performance. To address this, they proposed a group-wise expert allocation strategy (MoLA), which divides all layers into four groups and assigns varying numbers of experts to each group, ensuring that layers within the same group share the same number of experts. Building on this, Qing et al. (2024) introduced a layer-wise allocation strategy (AlphaLoRA), which theoretically determines the expert numbers for each layer based on its training quality.

Despite these advancements, two critical limitations remain, as shown in Table 1: 1) These methods determine the expert number without considering the downstream task. This is problematic, as different tasks may have varying levels of complexity and specific needs, which should influence the optimal expert configuration (as supported by experiments in Appendix A); 2) These methods also overlook the intrinsic rank of LoRA experts, typically assigning the same rank to all LoRA experts. This uniformity leads to equivalent representational capacities across experts, causing them to capture similar information. Thus, LoRA-MoE struggles to handle diverse and complex inputs.

To address these limitations, we propose GuiLoMo, a fine-grained strategy for jointly allocating layer-wise expert numbers and ranks in LoRA-MoE based on bilevel optimization with GuidedSelection vectors. GuiLoMo operates in two steps: 1) Obtaining GuidedSelection Vectors (GSVs): Through an initial optimization, GSVs are learned to guide LoRA-MoE in selecting the optimal expert numbers and ranks tailored to both the model backbone and the downstream task; 2) Allocating Expert Numbers and Ranks: After the prior optimization, the optimized GSVs are used to allocate expert numbers and ranks for LoRA-MoE, followed by the final training phase.

To summarize, our contributions are as follows:

1) To further unlock the potential of LoRA-MoE, we propose GuiLoMo, a fine-grained layer-wise expert numbers and ranks allocation strategy based on proposed GuidedSelection Vectors.

2) We conduct extensive experiments on a wide range of tasks, including natural language understanding, question answering, and mathematical reasoning, demonstrating the effectiveness of GuiLoMo. For instance, GuiLoMo achieves an average 2.61% improvement on mathmatical reasoning tasks with LLaMA-2_{7B}. Further analysis confirms the effectiveness of GuidedSelection vectors in selecting optimal expert numbers and ranks.

3) We provide valuable insights into the relationship between expert numbers, ranks, and their assigned layers. For example, we observe that multi-head attention (MHA) benefits more from increased expert numbers and ranks in bottom and top layers, whereas feed-forward networks (FFN) only exhibit this behavior in middle and top layers.

2 Preliminary

LoRA-MoE Framework LoRA-MoE integrates multiple vanilla LoRA experts into each pre-trained LLM submodule. Vanilla LoRA (Hu et al., 2022) efficiently adapts large models to downstream tasks by lowering computational and memory costs. For a pre-trained weight matrix $\mathbf{W_0} \in \mathbb{R}^{m\times n}$, LoRA creates two low-rank trainable matrices $\mathbf{A}$ and $\mathbf{B}$, where $\mathbf{B} \in \mathbb{R}^{m\times r}$, $\mathbf{A} \in \mathbb{R}^{r\times n}$, where $r \ll min(m, n)$. During training, $\mathbf{W_0}$ remains fixed while $\mathbf{A}$ and $\mathbf{B}$ are updated via gradient descent. The output representation h is defined as follows:

$$\mathbf{h} = \mathbf{W_0}x + \mathbf{BA}x \tag{1}$$

Every traditional LoRA-MoE layer incorporates N LoRA experts. The forward pass through the layer can be formulated as:

$$\mathbf{h} = \mathbf{W}_0 x + \sum_{i=1}^{N} \mathbf{G}(x)_i \mathbf{B_i A_i} x \tag{2}$$

where $\mathbf{G}(x) = \text{Softmax}(x\mathbf{W_r})$ represents the router in the LoRA-MoE layer. $\mathbf{W}_r$ is the trainable parameter matrix of the routing network that directs input x to different experts. By adaptively allocating inputs, the router promotes expert specialization, enhancing their ability to handle diverse tasks and input patterns.

Applying LoRA-MoE for LLMs LoRA-MoE is applied to key modules of LLMs, namely multi-head attention (MHA) and feed-forward networks (FFNs). In MHA, inputs are projected via $\mathbf{W}^Q$, $\mathbf{W}^K$, $\mathbf{W}^V$, and $\mathbf{W}^O \in \mathbb{R}^{d\times d}$. Each FFN uses gate- and up-projection matrices $\mathbf{W}^G$, $\mathbf{W}^U \in \mathbb{R}^{d\times d'}$, a activation (e.g., GELU), and a down-projection $\mathbf{W}^D \in \mathbb{R}^{d'\times d}$, where $d' > d$. GuiLoMo assigns optimal expert number and rank to these matrices.

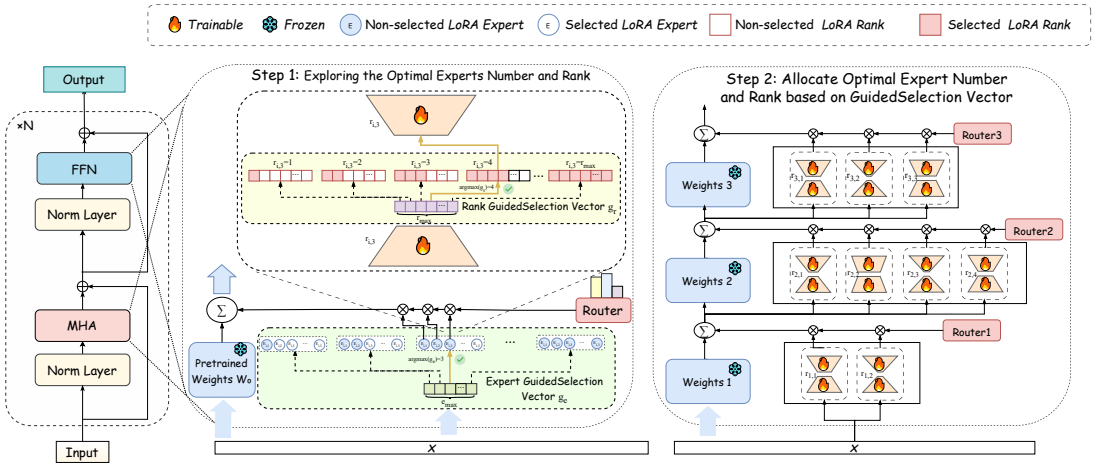


Figure 1: An illustration of our GuiLoMo strategy. GuiLoMo involves two steps: (Step 1): Exploring the optimal number of experts and ranks via a bilevel optimization algorithm with GuidedSelection Vectors. (Step 2): Allocate optimal expert number and rank based on GuidedSelection Vectors obtained in the previous step.

3 Method

In this section, we present our GuiLoMo strategy, which consists of two main steps: 1) A bilevel optimization algorithm is employed to obtain **GuidedSelection Vectors (GSVs)** of expert and rank for each module, tailored to the specific downstream task and model (§ 3.1); 2) Based on the obtained GSVs, the optimal expert number and rank for each module in LoRA-MoE are allocated, and the final training is then conducted (§ 3.4). See § 3.2 and § 3.3 for details of GSVs.

3.1 Bilevel Optimization for Obtaining the GuidedSelection Vector

In this section, we introduce the objective of bilevel optimization used to obtain **GuidedSelection Vectors** and its optimization process.

Optimization Objective Formally, our objective is to automatically determine the optimal expert number e_i^* for a given module (e.g., `down-projection in FFN`) within the i-th layer, and the optimal rank $r_{i,j}^*$ for the j-th expert under a specified LLM and downstream task setting.

To achieve this, we formulate the problem as an optimization task. In this process, we introduce the Expert GuidedSelection Vector $\mathbf{g}_E$ and Rank GuidedSelection Vector $\mathbf{g}_R$ as key components of the optimization, and the optimization objective is:

$$\min_{\{\mathbf{g}_E, \mathbf{g}_R\}} \mathcal{L}(\mathcal{D}, \pi_\theta, \mathbf{g}_E, \mathbf{g}_R) \tag{3}$$

$$\mathcal{L} = \mathcal{L}_{\text{SFT}} + \mathcal{L}_{\text{BAL}} \tag{4}$$

where π_θ is specific LLM and $\mathcal{L}_{\text{SFT}}$ denotes the supervised fine-tuning loss, which is computed via autoregressive language modeling on the downstream dataset $\mathcal{D}$, while $\mathcal{L}_{\text{BAL}}$ (refer to Eq. 16) represents the MoE balancing loss (Fedus et al., 2022; Zoph et al., 2022), which is introduced to encourage balanced utilization across experts and prevent expert collapse. The GuidedSelection Vector $\mathbf{g}_E \in \mathbb{R}^{e_{\max}}$ and $\mathbf{g}_R \in \mathbb{R}^{r_{\max}}$ are both trainable, with $e_{\max}$ and $r_{\max}$ representing the predefined maximum number of experts and ranks (see § 3.2 and § 3.3 for more details of $\mathbf{g}_E$ and $\mathbf{g}_R$). Since the optimization of $\mathbf{g}_E$, $\mathbf{g}_R$ should be under the optimal π_θ^*, we draw inspiration from Liu et al. (2019) and formulate the problem as a bilevel optimization:

$$\begin{aligned} &\min_{\{\mathbf{g}_E, \mathbf{g}_R\}} \mathcal{L}(\mathcal{D}_1, \pi_\theta^*, \mathbf{g}_E, \mathbf{g}_R) \\ &\text{s.t. } \pi_\theta^* = \arg\min_{\pi_\theta} \mathcal{L}(\mathcal{D}_2, \pi_\theta, \mathbf{g}_E, \mathbf{g}_R) \end{aligned} \tag{5}$$

where $\mathcal{D}_1$ and $\mathcal{D}_2$ are two splits of the training set $\mathcal{D}$ with equal size.

Optimization Process Based on the above objective, we formulate the overall procedure for obtaining the optimized GSVs in only a few T training steps. For a specific t-th training step, we first obtain $\pi_\theta^*(t)$ following Eq. 6, and then optimize the GSVs $\mathbf{g}_{E/R}^{(t)} = \{\mathbf{g}_E, \mathbf{g}_R\}^{(t)}$ with $\pi_\theta^*(t)$ to obtain $\mathbf{g}_{E/R}^{(t+1)} = \{\mathbf{g}_E, \mathbf{g}_R\}^{(t+1)}$ following Eq. 7.[1] Finally, we use $\mathbf{g}_{E/R}^{(t+1)}$ to obtain $\pi_\theta(t+1)$ for next step following Eq. 8.

[1] T is a hyperparameter in our experiments.

$$\pi_\theta^*(t) = \pi_\theta(t) - \xi_\theta * \nabla_{\pi_\theta(t)} \mathcal{L}(\mathcal{D}_2, \pi_\theta(t), \mathbf{g}_E^{(t)}, \mathbf{g}_R^{(t)}) \quad (6)$$

$$\mathbf{g}_{E/R}^{(t+1)} = \mathbf{g}_{E/R}^{(t)} - \xi_\mathbf{g} * \hat{\nabla}_{\mathbf{g}_{E/R}^{(t)}} \mathcal{L}(\mathcal{D}_1, \pi_\theta^*(t), \mathbf{g}_E^{(t)}, \mathbf{g}_R^{(t)}) \quad (7)$$

$$\pi_\theta(t+1) = \pi_\theta(t) - \xi_\theta * \nabla_{\pi_\theta(t)} \mathcal{L}(\mathcal{D}_2, \pi_\theta(t), \mathbf{g}_E^{(t+1)}, \mathbf{g}_R^{(t+1)}) \quad (8)$$

where ξ_θ and $\xi_\mathbf{g}$ are the learning rate for updating LLM weights and GSVs, respectively. $\hat{\nabla}$ indicates that we apply the STGE technique to ensure proper gradient flow (See § 3.2 and § 3.3 for more details). The overall optimization procedure is summarized in Alg. 1. The final obtained $\mathbf{g}_E^*$ and $\mathbf{g}_R^*$ determine the optimal number of experts and ranks according to the strategy described in § 3.4. GuiLoMo progressively learns the optimal heterogeneous LoRA-MoE configuration, allowing it to meet model- and task-specific needs.

Algorithm 1: Optimization Process

Input: Predefined maximum number of experts e_{max} and LoRA rank r_{max} per module, T optimization steps, learning rates ξ_θ and $\xi_\mathbf{g}$ for Model weights and GSVs.

Output: The optimized Expert GSV $\mathbf{g}_E^*$ and Rank GSV $\mathbf{g}_R^*$.

Initialize the LoRA-MoE framework according to the e_{max} and r_{max};
Split the training set $\mathcal{D}$ into $\mathcal{D}_1$ and $\mathcal{D}_2$;
for $t = 0$; $t < T$ **do**
  Obtain LLM weight $\pi_\theta^*(t)$ using Eq. 6 with learning rate ξ_θ;
  Obtain $\mathbf{g}_E^{(t+1)}$ and $\mathbf{g}_R^{(t+1)}$ using Eq. 7 with the gradients obtained from Eq. 11 and the learning rate $\xi_\mathbf{g}$;
  Obtain LLM weight $\pi_\theta(t+1)$ using Eq. 8 with the learning rate ξ_θ;
Derive the optimized Expert GSV $\mathbf{g}_E^*$ and Rank GSV $\mathbf{g}_R^*$.

3.2 Expert GuidedSelection Vector

For the Expert GSVs $\mathbf{g}_E \in \mathbb{R}^{e_{\text{max}}}$, we first predefine the maximum expert number e_{max} and initialize them with Gaussian distribution:

$$\mathbf{g}_E = \text{Softmax}(\boldsymbol{\alpha}), \quad with\ \boldsymbol{\alpha} = \{\alpha_i\}_{i=1}^{e_{\text{max}}} \quad (9)$$

where $\alpha_i \sim \mathcal{N}(0,1)$, and $\mathbf{g}_E$ denotes the selection probabilities for different allocated expert number settings. GuiLoMo selects the expert number setting by taking the index of the maximum value in $\mathbf{g}_E$. For example, if the maximum value of $\mathbf{g}_E^i$ at the i-th layer occurs at the 3-th position during the current training step, we allocate 3 experts for this module (see the green region in Fig. 1). Since $\mathbf{g}_E^i$ is learned through a few optimization steps on the task-specific data, the expert selection process described above needs to be differentiable. To guarantee gradient flow and enable end-to-end optimization, we adopt the Straight-Through Gradient Estimator (STGE) (Bengio et al., 2013) along with an auxiliary virtual vector $\mathcal{M}_E$ to approximate discrete selection while maintaining differentiability. Let $n^\star$ denote the index of the maximum value in $\mathbf{g}_E$. The forward propagation of the expert virtual vector $\mathcal{M}_E \in \{0, -\infty\}^{e_{\text{max}}}$ is formulated as follows:

$$\mathcal{M}_E^i = \begin{cases} 0, & \text{if } i \leq n^\star \\ -\infty, & \text{if } i > n^\star \end{cases} \quad (10)$$

For example, when allocating 3 experts, the expert virtual vector $\mathcal{M}_E$ is: $[0, 0, 0, -\infty, ..., -\infty]$. Meanwhile, in the backward propagation, we propagate the gradient flow from $\mathcal{M}_E$ to $\mathbf{g}_E$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{g}_E} = \mathcal{H}\left(\frac{\partial \mathcal{L}}{\partial \mathcal{M}_E}\right) \quad (11)$$

For more details on the $\mathcal{H}$ operation, please refer to Appendix G. The $\mathcal{M}_E$ is applied to top-K routing process to guide the learning of $\mathbf{g}_E$:

$$\hat{G}(x) = \frac{\text{TopK}(\text{Softmax}(x\mathbf{W}_r + \mathcal{M}_E), K)_i}{\sum_{i=1}^{K} \text{TopK}(\text{Softmax}(x\mathbf{W}_r + \mathcal{M}_E), K)_i} \quad (12)$$

where $\mathbf{W}_r$ denotes the weight of routing network.

3.3 Rank GuidedSelection Vector

The Rank GSVs $\mathbf{g}_{R \in \mathbb{R}^{r\text{max}}}$ shares a similar concept with the Expert GSVs during bilevel optimization. It begins by predefining the maximum rank r_{max} and is also initialized with Gaussian distribution using Eq. 9. However, the semantic meaning of each element differs, where each element in $\mathbf{g}_R$ represents a specific rank assigned to the corresponding expert. We select the index of maximum value in $\mathbf{g}_R$, i.e., $m^\star$, to determine the rank for the current training step. Similar to $\mathbf{g}_E$, $\mathbf{g}_R$ is non-differentiable during this process; therefore, we design rank virtual vector $\mathcal{M}_R \in \{0, 1\}^{r_{\text{max}}}$ to address this issue:

$$\mathcal{M}_R^i = \begin{cases} 1, & \text{if } i \leq m^\star \\ 0, & \text{if } i > m^\star \end{cases} \quad (13)$$

Models	Strategy	MRPC	COLA	RTE	ScienceQA	CommonsenseQA	OpenBookQA	Avg.
LLaMA$_{7B}$	MoLA(5)-Uniform(8)	82.43	84.18	83.03	90.28	75.10	76.00	81.84
	AlphaLoRA-Uniform(8)	85.19	85.42	85.19	90.37	76.49	78.20	83.48
	MoLA(5) + SoRA	82.55	84.76	83.03	90.38	75.35	76.80	82.15
	AlphaLoRA + SoRA	**85.51**	85.62	85.20	90.78	76.82	78.20	83.69
	GuiLoMo (Ours)	85.04	**85.71**	**85.92**	**91.50**	**77.15**	**78.60**	**83.99**
LLaMA-2$_{7B}$	MoLA(5)-Uniform(8)	84.17	86.19	84.83	92.08	77.55	80.00	84.14
	AlphaLoRA-Uniform(8)	84.23	86.67	**87.36**	92.71	78.05	80.80	84.97
	MoLA(5) + SoRA	84.46	86.31	84.84	92.36	77.81	80.20	84.31
	AlphaLoRA + SoRA	84.99	85.81	87.00	92.31	78.38	80.00	84.75
	GuiLoMo (Ours)	**85.80**	**87.25**	**87.36**	**92.99**	**78.46**	**81.20**	**85.51**
LLaMA-3$_{8B}$	MoLA(5)–Uniform(8)	86.61	87.15	87.73	93.97	79.52	83.40	86.40
	AlphaLoRA-Uniform(8)	87.13	88.88	88.09	94.42	80.02	83.80	87.06
	MoLA(5) + SoRA	85.97	87.54	88.45	94.24	79.44	84.00	86.61
	AlphaLoRA + SoRA	87.07	88.69	**89.53**	94.20	80.18	84.00	87.28
	GuiLoMo (Ours)	**87.77**	**89.26**	88.45	**94.83**	**81.24**	**85.60**	**87.86**

Table 2: Accuracy comparison of different methods under direct fine-tuning for each dataset. MoLA(5) indicates assigning a uniform 5 experts to each layer. Uniform(8) denotes setting all the rank of LoRA expert to 8.

For example, if the maximum value of $\mathbf{g}_R$ at a given training step is located at the 4-th element, the rank for this module is set to 4 (see the yellow region in Fig. 1). Accordingly, the corresponding Rank GuidedSelection Vector $\mathcal{M}_R$ is $[1, 1, 1, 1, 0, ..., 0]$.

Then, we parameterize on each LoRA expert matrix, denoted as $\Delta = \mathbf{BA} \in \mathbb{R}^{m \times n}$ (Eq. 1), in a form that mimic singular value decomposition (SVD) to obtain $\Delta = \mathbf{P\Lambda Q}$. $\mathbf{P} \in \mathbb{R}^{d_1 \times r_{\text{max}}}$ and $\mathbf{Q} \in \mathbb{R}^{r_{\text{max}} \times d_2}$ correspond to the original LoRA matrices $\mathbf{B}$ and $\mathbf{A}$, respectively, and $\mathbf{\Lambda}$ are initialized to 1. Note that we do not perform exact SVD. Subsequently, the rank virtual vector $\mathcal{M}_R$ is integrated with $\mathbf{\Lambda}$ and is incorporated into Eq. 2 to perform forward propagation:

$$\mathbf{h}' = \mathbf{W}_0 x + \sum_{i=1}^{K} \hat{\mathbf{G}}(x)_i \mathbf{P}(\mathcal{M}_R \odot \mathbf{\Lambda} \odot \mathbf{Q}x) \quad (14)$$

where $\odot$ denotes element-wise dot product, and $\hat{\mathbf{G}}$ is defined in Eq. 12. $\mathcal{M}_R$ guide the learning of $\mathbf{g}_R$, and its gradients are backpropagated in the same manner as $\mathcal{M}_E$ in Eq. 11, using STGE technique.

3.4 Allocating Expert Number and Rank via GSV

After obtaining optimized expert and rank GSVs, i.e., $\mathbf{g}_E^*$ and $\mathbf{g}_R^*$, the optimal expert number e^* and rank r^* are determined by selecting the index corresponding to the maximum values:

$$\begin{aligned} e_i^* &= \text{argmax}\ (\mathbf{g}_E^* i) \\ r_{i,j}^* &= \text{argmax}\ (\mathbf{g}_R^{*i,j}) \end{aligned} \quad (15)$$

where $e_i^* \leq e_{\text{max}}$ and $r_{i,j}^* \leq r_{\text{max}}$ denote the assigned expert number and rank in the i-th layer and the rank of the j-th expert in the i-th layer, respectively. Subsequently, we fine-tune the model using the loss function defined in Eq. 4 with expert number e^* and rank r^*, where the LoRA-MoE weights are initialized with $\pi_\theta^*(T)$.

4 Experiment

In this section, we conduct extensive experiments to examine the performance of GuiLoMo. We also conduct extra experimental analyses to gain deeper insights into this field, as presented in § 4.3. Implementation details can be found in Appendix D.

4.1 Experimental Settings

Datasets Following Qing et al. (2024), we evaluate our model on three natural-language understanding (NLU) tasks from GLUE: (1) the Microsoft Research Paraphrase Corpus (MRPC) (Dolan and Brockett, 2005), (2) the Recognizing Textual Entailment (RTE) dataset (Wang et al., 2019), (3) the Corpus of Linguistic Acceptability (CoLA) (Wang et al., 2019), and three reasoning-focused question-answering (QA) tasks: (1) ScienceQA (Lu et al., 2022), (2) CommonsenseQA (Talmor et al., 2019), and (3) OpenBookQA (Mihaylov et al., 2018).

We also evaluate GuiLoMo on mathematical reasoning benchmarks. Specifically, we perform instruction tuning on the MetaMathQA (Yu et al., 2024) dataset and evaluate on three benchmarks: (1) MultiArith (Roy et al., 2015), (2) SVAMP (Patel et al., 2021), and (3) GSM8K (Cobbe et al., 2021).

Models	Strategy	GSM8K	SVAMP	MultiArith	Avg.
$LLaMA_{7B}$	M(5)-U(8)	44.04	52.00	88.16	61.40
	A-U(8)	45.03	53.60	86.67	61.77
	M(5) + S	43.97	53.80	88.50	62.09
	A + S	46.10	54.80	89.17	63.36
	GuiLoMo (Ours)	**47.01**	**56.60**	**91.17**	**64.93**
$LLaMA\text{-}2_{7B}$	M(5)-U(8)	49.50	57.10	87.00	64.53
	A-U(8)	50.42	57.00	91.33	66.25
	M(5) + S	50.42	57.90	88.33	65.55
	A + S	51.48	58.00	92.50	67.33
	GuiLoMo (Ours)	**53.07**	**59.20**	**93.67**	**68.65**
$LLaMA\text{-}3_{8B}$	M(5)-U(8)	71.03	74.30	96.83	80.72
	A-U(8)	71.49	75.10	97.33	81.31
	M(5) + S	71.72	73.50	97.00	80.74
	A + S	**73.01**	75.30	97.33	81.88
	GuiLoMo (Ours)	72.85	**76.00**	**97.83**	**82.23**

Table 3: The results of mathematical reasoning under three models. M(5)-U(8) denotes MoLA(5)–Uniform(8); A-U(8) denotes AlphaLoRA-Uniform(8); M(5) + S denotes MoLA(5) + SoRA; A +S: AlphaLoRA + SoRA. MoLA(5) indicates assigning a uniform 5 experts to each layer. Uniform(8) represents setting all the rank of LoRA expert to 8.

See Appendix B for the detailed statistics of all the datasets used in our experiments.

Models We have applied our method to $LLaMA_{7B}$ (Touvron et al., 2023a), $LLaMA\text{-}2_{7B}$ (Touvron et al., 2023b), $LLaMA\text{-}3_{8B}$ (Cobbe et al., 2021), and $Mistral\text{-}v0.1_{7B}$ (Jiang et al., 2023).

Baselines We compare our GuiLoMo strategy with current state-of-the-art (SOTA) methods including MoLA (Gao et al., 2024), AlphaLoRA (Qing et al., 2024), MoLA+SoRA, and AlphaLoRA+SoRA. SoRA (Ding et al., 2023) is a variant of LoRA that allows for dynamic adjustments to the intrinsic rank during the adaptation process.[2] Implementation details of baselines can be found in Appendix E.

4.2 Main Result

Table 2 reports the results on three NLU tasks and three QA benchmarks. Across these datasets, GuiLoMo surpasses every baseline in terms of Avg. performance. Specifically, relative to AlphaLoRA-Uniform(8)[3], GuiLoMo delivers consistent gains of 0.61%, 0.64%, and 0.84% on the three model settings, respectively. GuiLoMo also outperform baselines on mathematical reasoning task. As show in Table 3, GuiLoMo exceeds AlphaLoRA + SoRA by an average of of 2.48%, 2.61%, and 0.43% on $LLaMA_{7B}$, $LLaMA\text{-}2_{7B}$, and $LLaMA\text{-}3_{8B}$, respectively. Based on these observations, we conclude that: *GuiLoMo, which flexibly allocates expert number and rank tailored to both model- and task-specific demands, further unleashes the potential of LoRA-MoE and leads to improved performance.*

[2]We adopt SoRA as it represents the current SOTA among LoRA-based methods that enable dynamic adjustments to the intrinsic rank during the adaptation process.

[3]"Uniform(8)" refers to assigning the same rank 8 to all LoRA experts.

Strategy	Avg.
MoLA(5)-Uniform(8)	79.42
GuiLoMo (Ours)	**81.87**
w/o adaptive expert allocation	80.64
w/o varying rank	80.97

Table 4: Average results of ablation studies on GuiLoMo across six tasks. MoLA(5) indicates assigning a uniform 5 experts to each layer. Uniform(8) represents setting all the rank of LoRA expert to 8. See Table 10 for detailed results. "w/o" means the exclusion of this strategy from GuiLoMo.

4.3 Further Analysis

Ablation Study of GuiLoMo Strategy We conduct ablation studies to assess the effectiveness of GuiLoMo with $LLaMA\text{-}2_{7B}$ across NLU and QA benchmarks on two different settings: (1) a fixed uniformly distributed number of experts with varying ranks, (2) a fixed uniformly assigned rank with varying expert allocation. As shown in Table 4, compared with the uniformly-allocated baseline MoLA(5)-Uniform(8), applying GuiLoMo exclusively to expert allocation or exclusively to rank allocation results in average performance improvements of 1.95% and 1.53%, respectively. The results also show that excluding either expert allocation or rank allocation from GuiLoMo leads to performance drops of 1.50% and 1.10%, respectively. Accordingly, we highlight the following insight:

> **Insight 1.** Jointly optimizing both expert and rank allocations outperforms optimizing either one in isolation.

Results across Model Families and Scales We conduct extra experiments on another family model $Mistral\text{-}v0.1_{7B}$ and larger-scale model $LLaMA\text{-}2_{13B}$ across three benchmarks to examine the generalization of our GuiLoMo. As shown in Table 5, GuiLoMo achieves average score improvements of 0.79% and 0.18% over the AlphaLoRA+SoRA on $LLaMA\text{-}2_{13B}$ and

Models	Strategy	MRPC	COLA	ComQA	Avg.
LLaMA-2_{13B}	MoLA(5)–Uniform(8)	86.78	87.82	81.74	85.45
	AlphaLoRA + SoRA	87.13	88.97	83.21	86.44
	GuiLoMo (Ours)	**88.06**	**89.36**	**83.95**	**87.12**
Mistral-v0.1_{7B}	MoLA(5)–Uniform(8)	86.43	87.24	82.96	85.54
	AlphaLoRA + SoRA	88.00	**89.26**	83.87	87.04
	GuiLoMo (Ours)	**88.23**	89.17	**84.19**	**87.20**

Table 5: The scores on MRPC, COLA, and ComQA under the Mistral-v0.1_{7B} and LLaMA-2_{13B} models. Avg.: the average score over these three benchmarks.

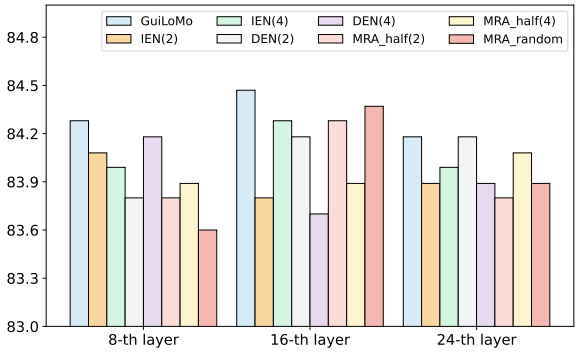


Figure 2: A comparative study of perturbed expert number e^* and rank r^* at different layers (8-th, 16-th, and 24-th). IEN(*) and DEN(*) denote the addition and removal of * experts, respectively. MRA_half(*): Half of the LoRA experts have their ranks increased by *, while the other half have their ranks decreased by * accordingly. MRA_random: Randomly shuffling the ranks of LoRA experts.

Mistral-v0.1_{7B}, respectively. The results further validate the widespread effectiveness of GuiLoMo across models of different scales and families.

Effectiveness of the Expert Number and Rank Assigned by GuiLoMo To validate the effectiveness of expert number e^* and rank r^* assigned by GuiLoMo that is tailored to specific models and tasks, we additionally conduct experiments with the following three strategies using LLaMA-2_{7B} on COLA benchmark. **1) Increase in Expert Number (IEN)** , increasing the number of experts while keeping the total rank ($\sum_{i=1}^{N}\sum_{j=1}^{e_i^*} r_{i,j}^*$) constant; **2) Decrease in Expert Number (DEN)**: Decreasing the number of experts while keeping the total rank constant [4]; **3) Mixed Rank Adjustment (MRA)** : Keeping the number of experts fixed, we randomly reassign ranks while keeping the total rank unchanged.

Note that only the expert number and rank of the specific m-th layer are intervened using the above three strategies, while the expert number and rank of the remaining layers remain unchanged (allocated by GuiLoMo). We apply these strategies to three layers (8, 16, 24) and report the results in Fig. 2. The results show that GuiLoMo outperforms all modified configurations, achieving the highest overall performance. From the results, we distill the following insight:

[4]To maintain a constant total rank in the LoRA-MoE framework, we proportionally reduce (or increase) the rank r^* previously assigned by GuiLoMo to each individual LoRA expert when total number of experts is increased (or decreased).

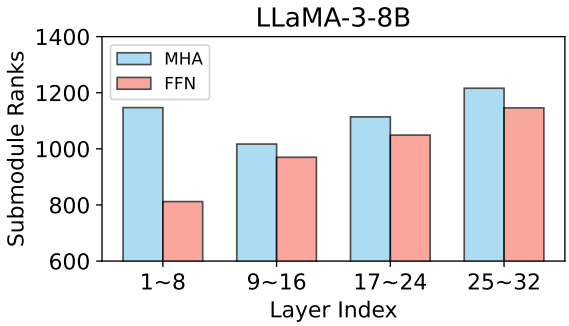


Figure 3: Total Rank of sub-modules (MHA and FFN) across different layer ranges in LLaMA-3_{8B} on CommonsenseQA.

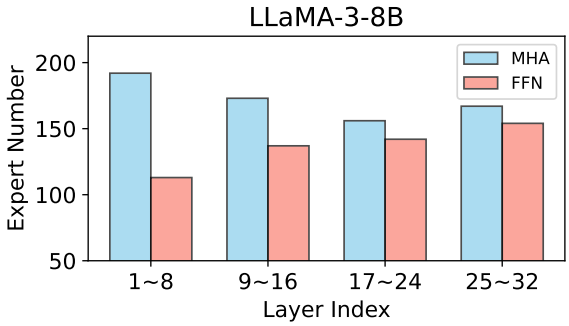


Figure 4: Total number of allocated experts for sub-modules (MHA and FFN) across different layer ranges in LLaMA-3_{8B} on CommonsenseQA.

> **Insight 2.** GuiLoMo allocates layer-wise optimal expert number and rank, better exploiting the potential of LoRA-MoE.

Allocation for MHA and FFN To delve deeper, we also observe the allocation patterns for MHA and FFN separately. We report the total assigned rank and average number of allocated experts for MHA and FFN under different layer ranges in Fig. 3 and Fig. 4, respectively. For example, the total rank (Total Rank of Submodules $= (\sum_{i=1}^{8}\sum_{j=1}^{e_i^*} r_{i,j}^*)$) in layer range $1 \sim 8$ of FFN, which includes gate-, up-, and down-projection. Based on Fig. 3 and Fig. 4, we draw the conclusion (see similar observations on other models and tasks in Appendix J):

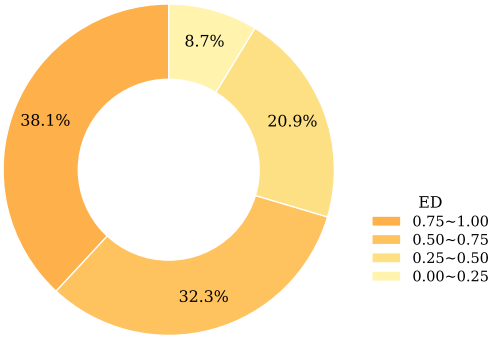


Figure 5: Distribution of ED scores computed by all the modules on LLaMA$_{7B}$, LLaMA-2$_{7B}$, and LLaMA-3$_{8B}$ under three NLU tasks.

Insight 3. MHA requires more expert numbers and ranks in bottom and top layers, whereas FFN shows this trend mainly in the middle and top layers.

Expert diversity We also explore Expert Diversity (ED) by quantifying it as the ratio between the size of the largest subset of experts whose ranks are all mutually distinct and the total number of experts (ED = |largest rank-distinct subset| / |all experts|).

For example, consider the FFN's up-projection module, which contains five experts with ranks $[3, 5, 6, 3, 7]$, so the expert diversity score ED $= 4/5 = 0.8$. In Fig. 5, we analyze the ED score for each submodule across NLU benchmarks on LLaMA$_{7B}$, LLaMA-2$_{7B}$, and LLaMA-3$_{8B}$. The results show that 38.1% of the ED scores fall within the high range of $0.75 \sim 1.00$, whereas only 8.7% are in the low range of $0.00 \sim 0.25$. Based on this observation, we draw the following conclusion:

Insight 4. Allocating diverse expert ranks enables more flexible and specialized adaptation to different tasks.

Impact of Task Difficulty We aim to investigate how the expert number e^* and rank r^* derived by GuiLoMo differ when facing challenging tasks compared to simpler ones. In pursuit of this goal, we use two BBH (Suzgun et al., 2023) sub-tasks, **Tracking Shuffled Objects** and **Logical Deduction**[5], each consists of sub-tasks differing in the number of objects K involved, with difficulty increasing as the number of objects grows.

[5]Given a set of K objects with initial positions and a sequence of pairwise swaps, determine their final positions after all transformations. Determine the sequential arrangement of K objects based on provided information regarding their relative positions and spatial relationships.

Task	Avg. Expert	Avg. Rank
Tracking shuffled objects		
— Object Number K = 3	5.87	6.13
— Object Number K = 5	6.13	5.98
— Object Number K = 7	6.35	6.07
Logical deduction		
— Object Number K = 3	5.23	6.44
— Object Number K = 5	5.41	6.51
— Object Number K = 7	5.76	6.40

Table 6: The average number of assigned experts across all modules ("module" here refers to all weight matrices where LoRA-MoE is applied, i.e., $\mathbf{W}^Q$, $\mathbf{W}^K$, $\mathbf{W}^V$, $\mathbf{W}^O$ in MHA and $\mathbf{W}^U$, $\mathbf{W}^D$, $\mathbf{W}^G$ in FFN) and the average rank across all experts, calculated under different object numbers within the same task.

From Table 6, we observe that as the number of objects increases, the number of experts assigned to different sub-tasks scales proportionally with the number of elements. However, the rank does not exhibit such a trend. Hence, we derive the following insight:

Insight 5. Within the LoRA-MoE, harder tasks benefit more from adding experts than from raising the rank of each LoRA expert.

5 Related work

LoRA-MoE Framework Recent research explores the integration of MoE (Shazeer et al., 2017) and LoRA (Hu et al., 2022), referred to as LoRA-MoE, to boost performance of LLMs in both single-task and multi-task scenarios in an efficient manner (Wu et al., 2024; Gao et al., 2024; Qing et al., 2024; Dou et al., 2024; Liu et al., 2023; Luo et al., 2024). For instance, Dou et al. (2024) leveraged LoRA-MoE to reduce catastrophic forgetting problem during supervised fine-tuning. Wu et al. (2024) uitlized LoRA-MoE with a hierarchical gating mechanism for efficient fusion across NLP and Vision & Language tasks. However, existing works merely allocate expert numbers and ranks uniformly for LoRA-MoE, failing to fully exploit its potential.

Allocation Strategy for LoRA-MoE To exploit the potential of LoRA-MoE, Gao et al. (2024) revealed that higher layers require more LoRA experts and initialized LoRA-MoE with different

numbers of experts with group-wise allocations. Moreover, Qing et al. (2024) leveraged Heavy-Tailed Self-Regularization (HT-SR) theory to develop a training-free and theoretically grounded method for allocating suitable expert numbers for LoRA-MoE. However, previous methods only consider the expert number while overlooking the expert rank, which results in all experts having the same capacity and thus lacking diversity. In contrast, our proposed GuiLoMo jointly optimizes both the expert number and rank.

6 Conclusion

In this work, we propose GuiLoMo, a fine-grained allocation strategy designed to fully exploit the potential of LoRA-MoE. GuiLoMo jointly determines expert number and rank through a bilevel optimization process. Unlike prior methods that rely on uniform or task-agnostic configurations, it introduces a GuidedSelection mechanism that guides the layer-wise allocation of expert number and rank in LoRA-MoE, tailored to both model-specific and task-specific needs. Extensive experiments demonstrate that GuiLoMo consistently improves model performance across a wide range of tasks. Furthermore, our analysis reveals how optimal expert configurations vary across layers and tasks, offering deeper insights into this field. We believe GuiLoMo paves the way for more flexible and efficient expert allocation strategies in future research.

Acknowledgement

This work is supported by the National Key R&D Program of China (2023YFC3304902). We would like to thank all the anonymous reviewers for their valuable comments and constructive feedback.

Limitations

While GuiLoMo demonstrates strong effectiveness and scalability in allocating expert number and rank for LoRA-MoE to both model-specific and task-specific settings, there remain two limitations. First, our experiments are limited to models up to 13B parameters, and we have not evaluated GuiLoMo on larger open-source LLMs such as LLaMA-70B due to computational constraints. Exploring its behavior on such super-sized models may provide further insights into scalability. Second, our evaluation is restricted to standard NLP tasks. It remains unclear whether GuiLoMo generalizes to other modalities or task types, such as cross-modal or multi-modal scenarios. We leave these directions for future work.

References

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Claudia Biancotti, Carolina Camassa, Andrea Coletta, Oliver Giudice, and Aldo Glielmo. 2024. Chat bankman-fried: an exploration of llm alignment in finance. *arXiv preprint arXiv:2411.11853*.

Yuan Chang, Ziyue Li, Hengyuan Zhang, Yuanbo Kong, Yanru Wu, Zhijiang Guo, and Ngai Wong. 2025. Treereview: A dynamic tree of questions framework for deep and efficient llm-based scientific peer review. *arXiv preprint arXiv:2506.07642*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems, 2021. *URL https://arxiv.org/abs/2110.14168*, 9.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. 2023. Sparse low-rank adaptation of pre-trained language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4133–4145, Singapore. Association for Computational Linguistics.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. LoRAMoE: Alleviating world knowledge forgetting in large language models via MoE-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1932–1945, Bangkok, Thailand. Association for Computational Linguistics.

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

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

Daniil Gurgurov, Tanja Bäumel, and Tatiana Anikina. 2024. Multilingual large language models and curse of multilinguality. *arXiv preprint arXiv:2406.10602*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Haoyang Huang, Tianyi Tang, Dongdong Zhang, Wayne Xin Zhao, Ting Song, Yan Xia, and Furu Wei. 2023. Not all languages are created equal in llms: Improving multilingual capability by cross-lingual-thought prompting. In *Findings of the Association for Computational Linguistics: EMNLP 2023*.

Yiren Jian, Tingkai Liu, Yunzhe Tao, Chunhui Zhang, Soroush Vosoughi, and Hongxia Yang. 2023. Expedited training of visual conditioned language generation via redundancy reduction. *arXiv preprint arXiv:2310.03291*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

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

Dawei Li, Shu Yang, Zhen Tan, Jae Baik, Sukwon Yun, Joseph Lee, Aaron Chacko, Bojian Hou, Duy Duong-Tran, Ying Ding, et al. 2024a. Dalk: Dynamic co-augmentation of llms and kg to answer alzheimer's disease questions with scientific literature. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 2187–2205.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, et al. 2025. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*.

Ziyue Li, Yuan Chang, and Xiaoqiu Le. 2024b. Simulating expert discussions with multi-agent for enhanced scientific problem solving. In *Proceedings of the Fourth Workshop on Scholarly Document Processing (SDP 2024)*, pages 243–256, Bangkok, Thailand. Association for Computational Linguistics.

Xiao Liang, Zhong-Zhi Li, Yeyun Gong, Yang Wang, Hengyuan Zhang, Yelong Shen, Ying Nian Wu, and Weizhu Chen. 2025. Sws: Self-aware weakness-driven problem synthesis in reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.08989*.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*.

Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. 2023. Moelora: An moe-based parameter efficient fine-tuning method for multi-task medical applications. *CoRR*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland. Association for Computational Linguistics.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *Advances in Neural Information Processing Systems*.

Tongxu Luo, Jiahe Lei, Fangyu Lei, Weihao Liu, Shizhu He, Jun Zhao, and Kang Liu. 2024. Moelora: Contrastive learning guided mixture of experts on parameter-efficient fine-tuning for large language models. *arXiv preprint arXiv:2402.12851*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple

math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Peijun Qing, Chongyang Gao, Yefan Zhou, Xingjian Diao, Yaoqing Yang, and Soroush Vosoughi. 2024. AlphaLoRA: Assigning LoRA experts based on layer training quality. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 20511–20523, Miami, Florida, USA. Association for Computational Linguistics.

Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3:1–13.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Saksham Sahai Srivastava and Ashutosh Gandhi. 2024. Mathdivide: Improved mathematical reasoning by large language models. *arXiv preprint arXiv:2405.13004*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

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

Yongqi Tong, Dawei Li, Sizhe Wang, Yujia Wang, Fei Teng, and Jingbo Shang. 2024. Can llms learn from previous mistakes? investigating llms' errors to boost for reasoning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3065–3080.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. 2020. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*.

Sizhe Wang, Yongqi Tong, Hengyuan Zhang, Dawei Li, Xin Zhang, and Tianlong Chen. 2024. Bpo: Towards balanced preference optimization between knowledge breadth and depth in alignment. *arXiv preprint arXiv:2411.10914*.

Xun Wu, Shaohan Huang, and Furu Wei. 2024. Mixture of loRA experts. In *The Twelfth International Conference on Learning Representations*.

Jing Xiong, Zixuan Li, Chuanyang Zheng, Zhijiang Guo, Yichun Yin, Enze Xie, Zhicheng Yang, Qingxing Cao, Haiming Wang, Xiongwei Han, et al. 2023. Dq-lore: Dual queries with low rank approximation re-ranking for in-context learning. *arXiv preprint arXiv:2310.02954*.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. 2023. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*.

Hang Yang, Hao Chen, Hui Guo, Yineng Chen, Ching-Sheng Lin, Shu Hu, Jinrong Hu, Xi Wu, and Xin Wang. 2024a. Llm-medqa: Enhancing medical question answering through case studies in large language models. *arXiv preprint arXiv:2501.05464*.

Shiping Yang, Jie Wu, Wenbiao Ding, Ning Wu, Shining Liang, Ming Gong, Hengyuan Zhang, and Dongmei Zhang. 2024b. Quantifying the robustness of retrieval-augmented language models against spurious features in grounding data. *arXiv preprint arXiv:2503.05587*.

Yaming Yang, Dilxat Muhtar, Yelong Shen, Yuefeng Zhan, Jianfeng Liu, Yujing Wang, Hao Sun, Weiwei Deng, Feng Sun, Qi Zhang, et al. 2025. Mtl-lora: Low-rank adaptation for multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 22010–22018.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations*.

Yiyao Yu, Yuxiang Zhang, Dongdong Zhang, Xiao Liang, Hengyuan Zhang, Xingxing Zhang, Ziyi Yang, Mahmoud Khademi, Hany Awadalla, Junjie Wang, et al. 2025. Chain-of-reasoning: Towards unified mathematical reasoning in large language models via a multi-paradigm perspective. *arXiv preprint arXiv:2501.11110*.

Hengyuan Zhang, Chenming Shang, Sizhe Wang, Dongdong Zhang, Feng Yao, Renliang Sun, Yiyao Yu, Yujiu Yang, and Furu Wei. 2024a. Shifcon: Enhancing non-dominant language capabilities with a shift-based contrastive framework. *arXiv preprint arXiv:2410.19453*.

Hengyuan Zhang, Yanru Wu, Dawei Li, Sak Yang, Rui Zhao, Yong Jiang, and Fei Tan. 2024b. Balancing speciality and versatility: a coarse to fine framework for supervised fine-tuning large language model. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 7467–7509.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*.

A Effect of Expert Number and Rank on Diverse Downstream Tasks

We design 6 allocation strategies to explore whether different tasks require different expert number and rank configurations. The strategies include three options for expert number and two options for rank, resulting in 6 combinations. Specifically, the expert number allocation includes two strategies from MoLA (Gao et al., 2024), and a normal allocation: MoLA(2468), MoLA(8642), and Gaussian distribution strategy (NormalE); the rank allocation strategies are remaining uniform (Uni), and Gaussian distribution strategy (NormalR). We set the total rank budget for each module to 40. NormalE selects 32 values of vertical coordinates from the standard normal distribution as selection probabilities, by uniformly sampling 32 input values within the interval $[-2\sigma, 2\sigma]$ and these values are then normalized and used to proportionally allocate the number of experts across layers, as illustrated in Fig. 6. NormalR follows the same allocation principle as NormalE, where ranks are proportionally assigned across layers based on a normalized standard normal distribution, given a total rank budget of 40 and predefined expert number per layer. For example, consider MoLA(2468)-Uni, where MoLA(2468) allocates 2 experts to each layer for the first 8 layers, 4 experts to each layer for 9-16 layers, 6 experts to each layer for 17-24 layers, and 8 experts to each layer for the last 8 layers. In the Uni setting, if the number of experts is 4, each expert is assigned a rank of $40 \div 4 = 10$. We conduct experiments on MRPC and ScienceQA datasets using $\text{LLaMA2}_{\text{7B}}$. The results are shown in Fig. 7. It can be observed that the performance varies across different expert number and rank configurations for the two tasks, demonstrating that different tasks require different expert numbers and ranks.

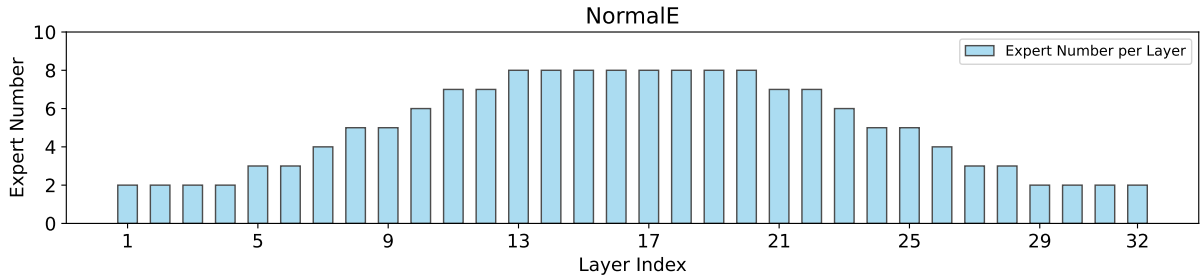


Figure 6: Allocating expert number across different layers using NormalE strategy.

B Dataset

The detailed statistics of all the datasets used in our experiments are reported in Table 7. We source each dataset from Huggingface Datasets and utilize the full dataset for our experiments.

Dataset	#Train	#Valid	#Test
CoLA	8,551	1,043	1,063
MRPC	3,668	408	1,725
RTE	2,490	277	3,000
ScienceQA	6,508	2,144	2,224
CommonsenseQA	9,740	1,221	1,140
OpenbookQA	5,957	500	500
MetaMathQA	394,999	-	-
MultiArith	420	-	180
SVAMP	700	-	300
GSM8K	7473	1319	-

Table 7: The detailed statistics of all the datasets we used in our experiments.

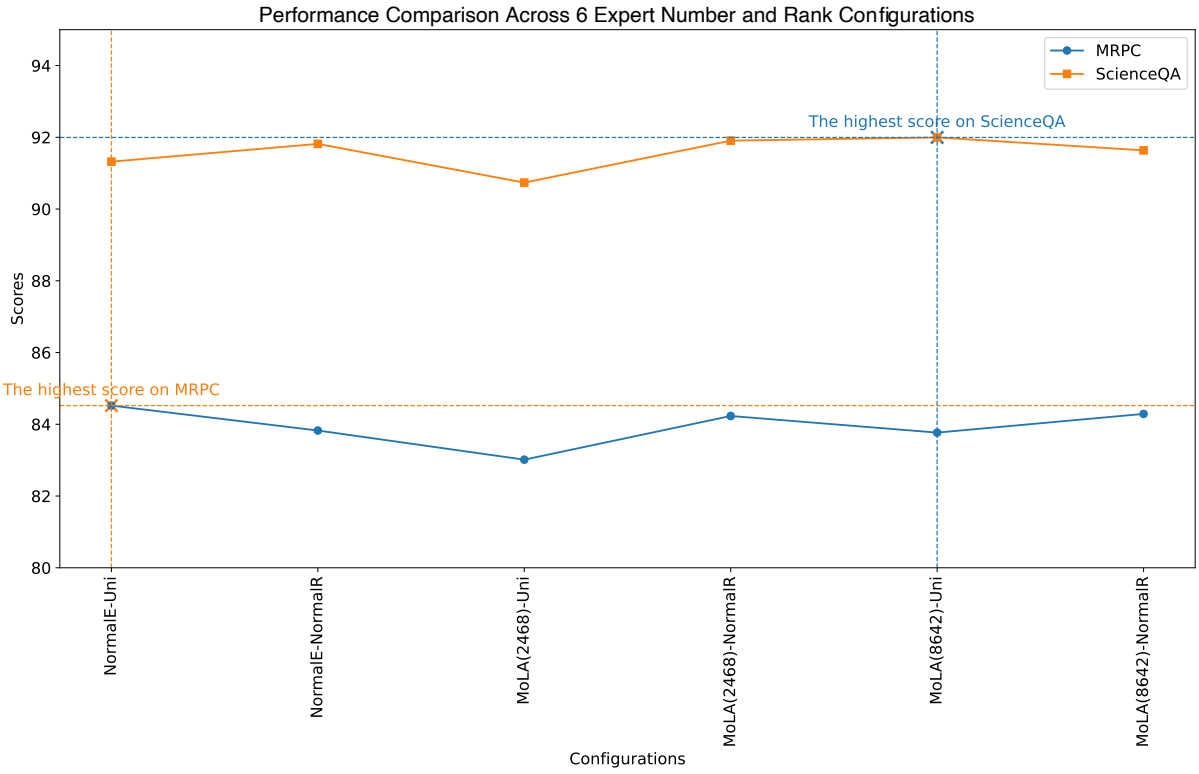


Figure 7: Performance comparison across 6 expert number and rank configurations in LLaMA-2_{7B} on MRPC and ScienceQA.

C Token Load Balance Loss

Consider a set of N experts indexed by $i = 1, ..., N$, and a batch of T tokens. The auxiliary loss is the scaled dot product of the expert usage vector f and routing probability vector P.

$$\mathcal{L}_{\text{BAL}} = c_B \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i \tag{16}$$

where f_i denotes the proportion of tokens assigned to expert i and P_i is the fraction of the router probability allocated for expert i. The auxiliary loss is scaled by $c_B = 10^{-3}$, which is large enough to encourage load balancing but small enough to avoid overshadowing the main objective.

D Implementation Details

All experiments are conducted on 8× NVIDIA A800-SXM4 80GB GPUs. The direct fine-tuning setting aligns with AlphaLoRA (Qing et al., 2024). We perform a grid search on the number of training epochs, including 10, 15, and 20 epochs for downstream task fine-tuning on the NLU and QA datasets. The cutoff length is set to 256 and the batch size is 32. For the mathematical reasoning tasks, we conduct instruction tuning on the MetaMathQA dataset (Yu et al., 2024) for 1 epoch with cutoff length set to 512. In GuiLoMo, we employ two separate AdamW optimizers: one for the GuidedSelection Vector (GSVs) and one for the trainable model parameters. In all experiments, we set $e_{\text{max}} = 8$, $r_{\text{max}} = 8$ and LoRA scale parameter α to 16. The optimizer for GSVs is configured with a learning rate of 3e-3, betas of (0.5, 0.999), a weight decay of 1e-3, and epsilon of 1e-8. The optimizer for the model parameters uses a learning rate of 3e-4, betas of (0.9, 0.999), and epsilon of 1e-8. We also employ a cosine learning rate scheduler to decay the learning rate. During the optimization process of Alg. 1, we trained for 3 epochs with a batch size of 64 on the NLU and QA datasets, and for 0.25 epoch with a batch size of 32 on the MetaMathQA dataset. T is computed as the dataset size divided by the batch size, multiplied by

the number of training epochs. Due to the intrinsic sparsity of GSV, we forgo the use of orthogonality regularization loss (Ding et al., 2023).

E The Configuration of the Baselines

We set $\beta = 2.5$ in AlphaLoRA and specify the total number of experts to be 160. The baselines are implemented using their open-sourced codes. For SoRA (Ding et al., 2023), we set the maximum decayed rank to 12, with $\lambda = 10^{-1}$, $\xi = 10^{-4}$, and $\eta_t = 10^{-1}$. MoLA(5) denotes using 5 experts per layer, while AlphaLoRA employs 160 in total. Under the Uniform(8) setting, each expert is assigned a rank of 8. When adopting the SoRA strategy for rank allocation, the maximum decayed rank is set to 12.

F Prompt Templates for Fine-tuning

We use the Alpaca prompt template for instruction tuning on three question answering datasets (ScienceQA, CommonsenseQA, and OpenbookQA):

```
Below is an instruction that describes a task, paired with an input that provides
further context. Write a response that appropriately completes the request.
### Instruction:
{instruction}
### Input:
{input}
### Response:
```

G $\mathcal{H}$ operation

We adopt the sensitivity (Zhang et al., 2023; Wang et al., 2020) without the norm to represent the discriminative score of the currently selected configuration:

$$\hat{S}(\phi) = \phi \nabla_{\phi} \mathcal{L} \tag{17}$$

where ϕ is any trainable parameter. Based on the above, the output of $\mathcal{H}(\phi)$ is such that at the position $n^{\star}$ (the index of the maximum value in ϕ), its value equals $\sum_{i=1}^{n^{\star}} \hat{S}(\phi)$, while all other positions are zero.

H Training Cost

We present GPU memory (GPU memory (GuiLoMo)) of LLaMA-2-7B for Stage 1 (GSV-based bilevel optimization). For comparison, we also recorded GPU memory (GPU memory (LoRA-MoE)) when directly applying LoRA-MoE without our GuiLoMo strategy (i.e., without GSV-based bilevel optimization). As shown in Table 8, the application of our GuiLoMo strategy, which includes GSV-based bilevel optimization, results in only a minimal increase in GPU memory usage.

	ScienceQA	OpenBookQA
GPU memory (GuiLoMo)	119.42GB	88.32GB
GPU memory (LoRA-MoE)	117.21GB	87.10GB

Table 8: GPU memory comparison with and without GuiLoMo strategy.

$(e_{\max}, r_{\max})$	(12,12)	(16,16)
GuiLoMo	93.21	93.30
AlphaLoRA + SoRA	92.67	92.85
MoLA-Uniform	92.13	92.31

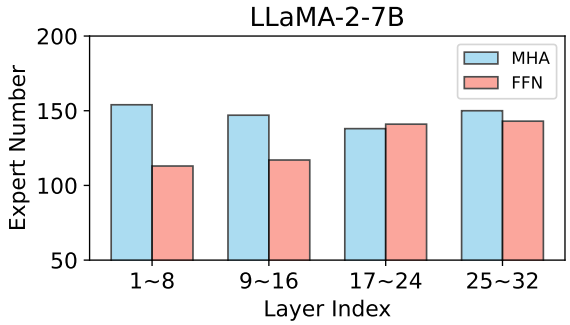
Table 9: Performance comparison of GuiLoMo with other models under different $(e_{\max}, r_{\max})$ settings.

I Sensitivity to Predefined Maximum Experts and Ranks

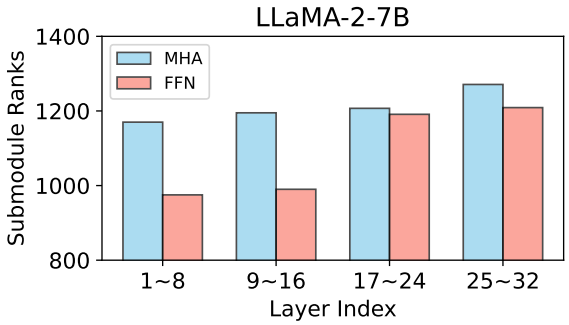
We evaluate GuiLoMo in two settings, i.e., $e_{max} = r_{max} = 12$, and $e_{max} = r_{max} = 16$, as for predefined maximum expert numbers and ranks. As shown in Table 9, under the same number of trainable parameters, GuiLoMo consistently outperforms other strong baselines, demonstrating its superior allocation capability and optimization efficiency.

J Allocation Details

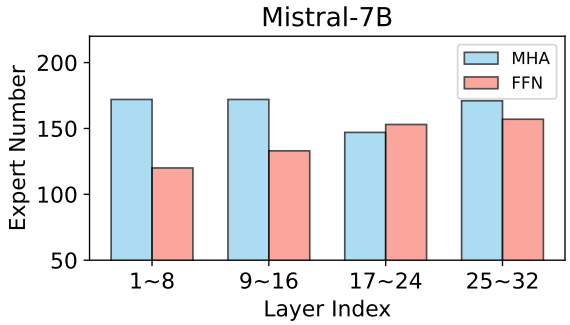
The total expert number and total rank of sub-modules (MHA and FFN) for LLaMA-2_{7B} and Mistral-v0.1_{7B} on MetaMathQA and COLA under different layer ranges are illustrated in Figs. 8(a), 8(b) and 8(c), 8(d), respectively.



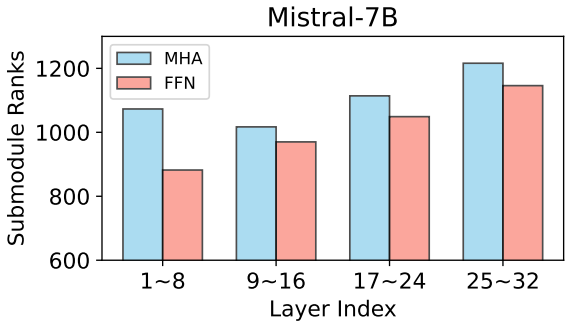
(a) Total number of allocated experts for sub-modules (MHA and FFN) across different layer ranges in LLaMA-2_{7B} on MetaMathQA.



(b) Total rank of sub-modules (MHA and FFN) across different layer ranges in LLaMA-2_{7B} on MetaMathQA.



(c) Total number of allocated experts for sub-modules (MHA and FFN) across different layer ranges in Mistral-v0.1_{7B} on COLA.



(d) Total rank of sub-modules (MHA and FFN) across different layer ranges in Mistral-v0.1_{7B} on COLA.

K Detailed Results of Ablation Studies

Strategy	MRPC	COLA	SciQA	ComQA	GSM8K	MultiArith	Avg.
MoLA(5)-Uniform(8)	84.17	86.19	92.08	77.55	49.50	87.00	79.42
GuiLoMo	**85.80**	**87.25**	**92.99**	78.46	**53.07**	**93.67**	**81.87**
w/o adaptive expert allocation	84.99	86.86	92.13	**78.54**	52.16	89.17	80.64
w/o varying rank	**85.80**	86.77	92.36	78.13	51.10	91.67	80.97

Table 10: The detailed results of ablation studies on GuiLoMo across from six benchmark (MRPC, COLA, SciQA, ComQA, GSM8K, MultiArith).