Rank Also Matters: Hierarchical Configuration for Mixture of Adapter Experts in LLM Fine-Tuning

Anonymous ACL submission

Abstract

Large language models (LLMs) have demonstrated remarkable success across various tasks, accompanied by a continuous increase in their parameter size. Parameter-Efficient Fine-Tuning (PEFT) methods, such as Low-Rank Adaptation (LoRA), address the challenges of fine-tuning LLMs by significantly reducing the size of trainable parameters. Recent studies have enhanced LoRA using Mixture of Experts (MoE) routing strategy, leveraging multiple adapter experts and gating mechanism to improve fine-tuning performance further. However, existing methods primarily adjust the number of adapter experts per layer while overlooking the adapter rank, which is a critical factor influencing both representational capacity and parameter size. To this end, we propose HILO, the first training-free configuration strategy that jointly allocates expert number and rank for mixture of adapter experts. Extensive experiments demonstrate that HILO achieves consistent Pareto improvements in the accuracy–efficiency trade-off, reducing both trainable and active parameters while maintaining or slightly improving performance.

1 Introduction

In recent years, the rapid advancement of the Large Language Model (LLM) has demonstrated impressive performance on various tasks, *e.g.*, natural language processing, computer vision, and multimodal applications (Liu et al., 2024b; Chang et al., 2024). However, as the parameter size of large models continues to scale up, substantial computational resources are required for model fine-tuning. In this context, Parameter-Efficient Fine-Tuning (PEFT) methods have emerged (Han et al., 2024), significantly reducing the number of trainable parameters while maintaining performance. PEFT techniques include adapter tuning, prompt tuning, prefix tuning, and Low-Rank Adaptation (LoRA) (Houlsby et al., 2019; Lester et al., 2021; Li and Liang, 2021; Hu et al., 2022). Among these methods, LoRA has attracted significant attention for its strategy of injecting low-rank adapter matrices into existing weight matrices, enabling efficient fine-tuning for downstream tasks while introducing a small amount of additional trainable parameters.

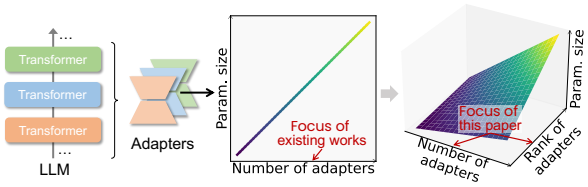


Figure 1: **Optimization focus of adapter experts**. Existing works primarily optimize the number of adapters, whereas this paper explores both the number and rank of adapters to improve fine-tuning performance.

Based on LoRA, several studies have further optimized it by incorporating the routing mechanism of Mixture of Experts (MoE) (Liu et al., 2024a; Zeng et al., 2024; Dou et al., 2024). MoE achieves superior computational efficiency and model capacity by employing multiple feed-forward networks (referred to as experts) to decompose complex problems into smaller subproblems, which are processed by a subset of experts selected by a gating network (Zhou et al., 2022). The integration of LoRA and such routing mechanism is widely termed the Mixture of Adapter Experts (MoAE), which involves introducing multiple adapters for the model's weight matrices and activating a subset of them selectively via a gating network, resulting in an efficient and scalable fine-tuning solution.

The vanilla MoAE approach typically assigns a uniform configuration of adapter experts across all layers. However, such a layer-agnostic strategy fails to account for the inherent variability in representational complexity across pre-trained model layers, leading to suboptimal utilization of the limited trainable parameter budget. In this context, some studies have attempted to refine the number of experts based on layer-specific characteristics. For

instance, (Gao et al., 2024) allocates fewer experts to shallow layers , while (Qing et al., 2024) adjusts expert counts based on layer training quality. Despite these advancements, existing frameworks still rely on a rigid, uniform rank for all adapters, overlooking the diverse representational requirements across different layers. As illustrated in Figure 1, beyond the expert count, the rank is also a critical dimension that governs both the representational capacity and the trainable parameter footprint of each adapter expert.

To this end, we present HILO, a training-free hierarchical configuration scheme designed to improve the parameter utilization efficiency of MoAE. Unlike prior studies that focus solely on the number of adapter experts, HILO optimizes both the numbers and ranks of adapter experts, tailoring adapter allocation along dual dimensions to accommodate the varying representational complexity across model layers. Experimental results demonstrate that HILO consistently improves the accuracy–efficiency trade-off without introducing additional training cost or search overhead, reducing both trainable and active parameters while maintaining or slightly improving performance. This design targets practical deployment scenarios where additional training or hyperparameter search is infeasible, requiring simple yet effective configuration strategies.

The contributions of this work are as follows:

- We demonstrate that the existing mixture of adapter experts architectures primarily focus on the number of experts, and our empirical results reveal that the adapter rank is also a critical factor.
- We propose HILO, a training-free hierarchical configuration scheme that jointly allocates diverse numbers and ranks of adapter experts across layers to improve the efficiency–performance trade-off under a fixed parameter budget.
- We conduct extensive experiments showing that HILO achieves higher accuracy while simultaneously reducing trainable and active parameters compared with existing methods.

2 Related Work

2.1 Parameter Efficient Fine-Tuning (PEFT)

The increasing size of LLM parameters motivates the development of parameter-efficient fine-tuning (Han et al., 2024). Unlike full-parameter fine-tuning, PEFT methods selectively adjust or introduce a small number of trainable parameters without modifying the entire parameter set. Adapter tuning (Houlsby et al., 2019), prompt tuning (Lester et al., 2021), prefix tuning (Li and Liang, 2021), and LoRA (Hu et al., 2022) are some representative PEFT schemes, and LoRA is one of the most widely employed strategies. It integrates the low-rank decomposition matrices into the weight update of the model and greatly reduces the number of trainable parameters. Specifically, the modified parameter matrix $w' \in \mathbb{R}^{n\times m}$ is computed as $w' = w + AB$, where $w \in \mathbb{R}^{n\times m}$ is the original parameter matrix, $A \in \mathbb{R}^{n\times r}$ and $B \in \mathbb{R}^{r\times m}$ are the low-rank decomposition matrices, and r is the rank value of the LoRA adapter, which is much smaller than n, *i.e.*, $r \ll min(n, m)$.

Building upon LoRA, several variants have been proposed to better match LLM characteristics and implementation constraints. QLoRA (Dettmers et al., 2024) reduces parameter overhead by quantizing adapter parameters, while LoRA+ (Hayou et al., 2024) improves fine-tuning performance by assigning differentiated learning rates to adapter matrices A and B. VeRA (Kopiczko et al., 2024) departs from conventional initialization by freezing randomly initialized A and B and introducing trainable vectors, further reducing trainable parameters at the cost of slight accuracy degradation. Other works on adaptive rank allocation, such as AdaLoRA (Zhang et al., 2025) and GeoLoRA (Schotthöfer et al., 2025), assume a single-adapter-per-layer formulation with a unified rank definition. In architectures with multiple gated adapter experts, no such unified rank exists due to expert-wise ranks and dynamic composition. Consequently, their underlying assumptions no longer hold, making these methods not directly transferable without fundamentally reformulating the problem. Accordingly, HILO focus on a plug-and-play, training-free configuration strategy.

2.2 Mixture of Experts

The mixture of experts architecture was first proposed by (Jacobs et al., 1991), introducing an assignment mechanism based on input data to distribute tasks across multiple expert modules, thereby achieving efficient task specialization and model capacity utilization. With the growing popularity of the Transformer (Vaswani, 2017), many studies revisited MoE by applying it to the cor-

responding Feed-Forward Network (FFN) layers, extending these layers into multiple expert networks. However, a prominent feature of the widely adopted MoE in transformer-based LLMs is the use of a sparse gating mechanism, which selects only a subset of experts for token processing, enabling LLMs to scale to an extreme scale.

GShard (Lepikhin et al., 2021) and Switch Transformer (Fedus et al., 2022) are pioneers that employ learnable top-2/top-1 expert selection strategies, and DeepSeek (Dai et al., 2024) implements a shared expert isolation scheme. HashLayer (Roller et al., 2021) uses a hashing-based way to select experts for tokens, improving the stability of model training. (Huang et al., 2024; Zhou et al., 2022; Yang et al., 2024b) allow different numbers of experts to be assigned for different tokens, enhancing model flexibility. Besides studies on architectures and training strategies of MoE, recent years have also witnessed the emergence of many MoE-based multimodal models (Mustafa et al., 2022).

2.3 LoRA Meets MoE

Based on basic LoRA architecture, some studies have further introduced the routing strategy of MoE, where the single LoRA adapter for each weight matrix is replaced by a set of adapter experts managed by a gating network (Yang et al., 2024a; Wu et al., 2024a). This approach enhances the capability of the fine-tuned models while maintaining scalability (Li et al., 2024; Wu et al., 2024b).

MoELoRA (Liu et al., 2024a) combines the strengths of these two techniques to achieve an efficient multi-task fine-tuning framework. AdaMoE (Zeng et al., 2024) ingeniously introduces additional null expert networks, enabling a learnable and dynamic selection of the experts. LoRAMoE (Dou et al., 2024) classifies all experts of each layer into two categories, assigning them to process either world knowledge or fine-tuning knowledge, thereby mitigating the issue of world knowledge forgetting of fine-tuning. MoLA (Gao et al., 2024) implements a layer-wise expert allocation strategy and demonstrates through experimental results that deeper networks require more experts than shallower ones.

3 Insights and Motivations

Parameter-efficient fine-tuning aims to maximize model representational capacity under a constrained parameter budget. Existing approaches

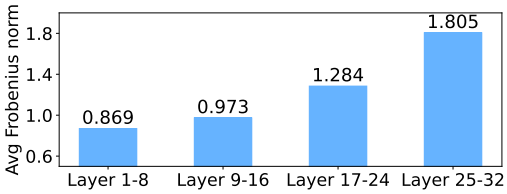


Figure 2: Frobenius-norm of adapters' parameter matrix

primarily focus on structural allocation. For instance, MoLA and AlphaLoRA (Gao et al., 2024; Qing et al., 2024) exploit the hierarchical nature of deep networks by assigning fewer experts to shallow layers and more to deeper layers, thereby improving efficiency from the perspective of expert number. However, these methods overlook another key factor, rank, that fundamentally determines the expressive power of each adapter.

From a theoretical perspective, the representational capacity of an adapter is governed by its low-rank structure. Given a weight matrix $W \in \mathbb{R}^{m\times n}$, its singular value decomposition is expressed as $W{=}\sum_{i=1}^{r} \sigma_i u_i v_i^T$. According to the Eckart-Young-Mirsky theorem, the optimal rank-k approximation is obtained by retaining the top-k singular values:

$$\min_{\text{rank}(W_k)\leq k} \|W - W_k\|_F^2 = \sum_{i=k+1}^{r} \sigma_i^2. \quad (1)$$

This indicates that singular values characterize the distribution of information energy, reflecting the importance of different representational directions.

Based on this insight, we fine-tuned a Llama 2-7B (Touvron et al., 2023) model on the ScienceQA dataset (Lu et al., 2022) using a vanilla MoAE architecture (8 experts per layer, rank of 8, and Top-2 activation). Then, we perform singular value decomposition and calculate the Frobenius-norm on adapter matrices across layers. As shown in Figure 2, the average Frobenius-norm value increases with layer-depth, indicating significant heterogeneity in feature energy across layers.

This observation suggests that adapter updates exhibit clear layer-wise heterogeneity. In particular, the Frobenius norm of adapter update matrices varies across depth, indicating that different layers do not use adapter capacity in the same way. Rather than implying a unique optimal rank schedule, this finding questions the common practice of assigning a uniform rank to all adapter experts. Since rank directly controls both the capacity and parameter cost of each expert, keeping it fixed across layers may limit the flexibility of parameter allocation in MoAE. That is, in addition to the number,

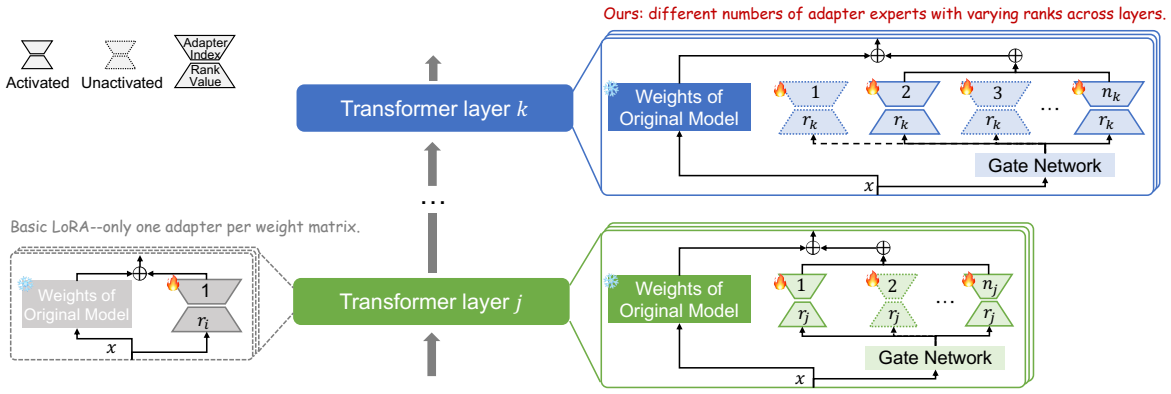


Figure 3: **HILO Design**. HILO configures number and ranks of adapter experts for weight matrices across different layers.

rank should also be included in the configuration of adapter experts.

Based on this insight, we extend the MoAE configuration space from a single-dimensional allocation over *expert number* to a two-dimensional allocation over both *number* and *rank* of expert. This allows the limited adapter budget to be distributed more flexibly across layers. We therefore propose HILO, a hierarchical expert configuration strategy that jointly adjusts the number and rank of adapter experts according to layer depth. To further verify the presence of layer-wise heterogeneity, we also analyze the output magnitudes of adapter experts across layers and obtain observations consistent with the Frobenius-norm analysis. Due to space limitations, the detailed results and discussion are provided in Appendix A.

This also motivates the need for a simple yet robust configuration strategy under practical constraints, rather than relying on complex adaptive schemes.

4 HILO Design

4.1 Mixture of Adapter Experts

Extending LoRA to MoE architectures involves transforming the original single adapter, which comprises one pair of low-rank matrices A and B, into multiple adapter experts. Suppose there is a model with N Transformer layers and each layer i has E_i adapter experts, denoted by e_i^j, where $j \in E_i$. In line with the basic LoRA initialization scheme, each adapter expert has its matrix A initialized with a random Gaussian distribution and B with zeros. Moreover, a gate network G_i is introduced for layer i. It is responsible for analyzing the features of each input token x and computing activation probabilities P_i^j for all adapter experts e_i^j via a softmax function.

Following the MoE computation procedure with a Top-K activation policy, for a given token x, the adapter experts $e_k^i, k \in K$ with the highest K probabilities are activated. Then, the output of this mixture of adapter experts layer can be expressed as:

$$Output_{e_i} = \sum_{k \in K} p_i^k \, w_i^k \, x, \quad p_i^k = \frac{P_i^k}{\sum_{j \in K} P_i^j},$$

where w_i^k represents the parameter matrix of the expert e_i^k. Assuming W_i is the original weight matrix of layer i, the final output of this layer yields: $Output_i = W_i \, x + Output_{e_i}$. So far, it is possible to select a subset of adapter experts dynamically, enhancing both the capacity and adaptation flexibility of LoRA. Meanwhile, $Output_{e_i}$ is fed into the next layer as the input.

4.2 Adapter Experts Setting

4.2.1 Dynamic Adapter Experts Allocation

The vanilla allocation strategy assigns the same number of adapter experts to each layer. However, as mentioned earlier, such an approach overlooks the differences across layers, resulting in inefficiencies such as introducing unnecessary trainable parameters or failing to utilize them fully. Two proven flexible expert allocation strategies are presented here (Gao et al., 2024; Qing et al., 2024). First, based on the varying complexity of processing tasks across different layers, experts can be allocated incrementally from shallow to deep layers. This intuitive method demonstrates performance improvements compared to a fixed expert allocation. Second, the Heavy-Tailed Self-Regularization theory can be used to evaluate the training quality of the original model layers, guiding a flexible and adaptive allocation of experts to each layer.

4.2.2 Activation of Adapter Experts

For expert activation, Top-K is a feasible and simple approach. However, activating a fixed number of experts neglects the processing complexity among different tokens, which may result in either redundancy or insufficiency for some tokens processing.

Comparatively, learning-based policies enable the activation of a variable number of experts. The Top-P series methods exemplify this policy by activating experts for each token, either by selecting in order of decreasing probability until the cumulative probability exceeds a predefined threshold P, or by directly selecting all experts whose probabilities surpass the threshold. P is predefined, and the loss function incorporates constraints on the number of activated experts. Additionally, another implementation involves inserting several placeholder adapters without parameters into the adapter expert layer (Zeng et al., 2024). These placeholders do not perform computations on the token but allow the model to activate a varying number of true experts for each token while still adhering to the Top-K policy. For instance, if n of the selected experts are placeholder adapters, then only $K - n$ true experts are actually activated. During training, the model learns to activate the appropriate number of experts dynamically for each token.

4.3 Hierarchical Rank Setting

Our design is guided by three practical constraints: (1) a training-free requirement that avoids additional optimization or hyperparameter search; (2) a low-granularity configuration space where rank values are restricted to a small discrete set (e.g., 2, 4, 6, 8); and (3) compatibility with existing MoAE frameworks without modifying their training pipeline.

In MoAE, both the number of adapter experts and their ranks critically influence model performance and parameter efficiency. While existing methods mainly focus on expert allocation, the rank directly determines the representational capacity and parameter cost of each expert. This suggests that assigning lower ranks to shallow layers and higher ranks to deeper layers can better match layer-wise representational demands and improve parameter utilization.

HILO aims to develop a simple yet effective and general hierarchical rank strategy. To this end, rather than introducing complex data-driven mechanisms to dynamically determine the rank of each expert, we adopt a monotonic linear rank allocation as a minimal yet effective strategy under the above constraints, which can be formally expressed as:

$$r_i = r_{min} + \left\lfloor \frac{r_{max} - r_{min}}{\lceil \frac{n}{l} \rceil} \right\rfloor \cdot \left\lfloor \frac{i-1}{l} \right\rfloor ,$$

where $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ respectively represent the rounding up and rounding down operations; r_{min} and r_{max} respectively denote the minimum and maximum rank values. Typically, r_{min} is set to 2 and r_{max} can take values such as 8, 16, 32, or higher depending on the base model scale, for instance, 8 is typical for 7B or 8B models (Gao et al., 2024; Qing et al., 2024; Zeng et al., 2024). n indicates the total layers of the fine-tuned model. Optionally, it is feasible to set the same rank for every l layers. Moreover, considering the computational characteristics of hardware devices, rank values can be flexibly configured as multiples or powers of 2 based on the formula in practical implementation, rather than strictly adhering to it.

Under the above constrained setting, more complex allocation strategies (e.g., non-monotonic or exponential schedules) would introduce additional design complexity while offering limited practical benefit. Therefore, a monotonic linear schedule serves as a robust default that balances simplicity, compatibility, and effectiveness. This method is training-free, aligning to minimize computational overhead in the context of PEFT. Moreover, its simplicity ensures seamless integration with existing frameworks. Extensive experimental results presented in §5 demonstrate that the design based on this assumption leads to improved performance and efficiency.

5 Evaluation

5.1 Experiment Setup

Models. Based on differences in network architectures and parameter scales, Llama 2-7B, Llama 3-8B, and Qwen 2-1.5B were selected for experimental evaluation. Given the widespread deployment and popularity of Llama 2-7B within the community (Touvron et al., 2023), it was mainly selected for ablation studies.

Datasets. To evaluate the performance of HILO and comparison methods, two distinct types of tasks are employed using widely recognized datasets. The first type of task involves commonsense reasoning, including the following datasets:

ScienceQA (SQA) (Lu et al., 2022), CommonsenseQA (CQA) (Talmor et al., 2019), and OpenBookQA (OQA) (Mihaylov et al., 2018). The second type of task focuses on semantic understanding, utilizing three datasets from the renowned GLUE Benchmark (Wang et al., 2019): Microsoft Research Paraphrase Corpus (MRPC), Recognizing Textual Entailment (RTE), and Corpus of Linguistic Acceptability (COLA).

Baselines. HILO is compared against several standard baselines and SOTA methods: 1) *Base model*: the original pre-trained model with prompt engineering serves as a basic baseline; 2)*Vanilla*: it allocates a fixed number of 8 adapter experts with a rank of 8 to every Transformer layer; 3) *MoLA* (Gao et al., 2024) employs an increasing allocation pattern from shallow to deep layers, assigning 2,4,6, and 8 experts to every 8 layers (denoted as ▽); 4)*AlphaLoRA* (Qing et al., 2024): it allocates experts based on the training quality of each layer (denoted as ‡); 5) *AdaMoE* (Zeng et al., 2024): it introduces placeholder experts to enable a learning-based expert activation, which uses a 1:1 ratio of real to placeholder experts in the experiments. For fair comparison, HILO is applied as an add-on to existing methods by retaining their expert allocation and activation strategies while integrating the proposed hierarchical rank setting.

5.2 Experiment Results

The trainable parameter volume introduced by vanilla, which mainly involves all adapters with the rank of 8, is defined as a unit of 1. The active parameters that the vanilla method activates 2 adapters with the rank of 8 for each token are defined as 2. Other methods are scaled proportionally based on the number of introduced or activated adapters and corresponding rank values. All experiments were performed on 4 NVIDIA 4090 GPUs.

We evaluate HILO primarily from an efficiency–performance trade-off perspective, rather than optimizing for accuracy alone. The experiments evaluate HILO from four perspectives: a) a fixed trainable-parameter budget, b) solely hierarchical rank setting, c) sensitivity analysis, and 4) resource overhead.

5.2.1 Budget-fixed Performance

For a fair comparison, this experiment adopts a fixed trainable-parameter budget, under which HILO reallocates the parameters saved from lower-rank shallow-layer adapters to additional adapter experts, thereby jointly configuring expert number and rank within the same budget. Except for the pre-trained base model and Vanilla, MoLA and AlphaLoRA have fixed trainable and active parameters, whereas AdaMoE has dynamic active parameters due to learned expert activation, leading to separate comparisons.

(1) HILO *vs*. MoLA & AlphaLoRA. MoLA reduces the total number of adapters in the model while keeping the adapter rank unchanged, thereby reducing the trainable parameters to 0.63 compared to Vanilla. To ensure fairness in the comparison, the trainable parameters introduced by AlphaLoRA are kept consistent with those of MoLA. Furthermore, since MoLA and AlphaLoRA also adopt the Top-2 activation scheme, their active parameters remain the same as Vanilla's.

Building on MoLA/AlphaLoRA, HILO progressively reduces rank values from shallow to deep layers to 2, 4, 6, and 8, grouped every 8 layers (denoted as ▽). To ensure fairness of comparison, the total trainable parameter size of HILO is kept consistent with MoLA and AlphaLoRA by proportionally increasing the number of adapter experts from shallow to deep layers. Compared to MoLA, HILO assigns eight experts per layer, while compared to AlphaLoRA, the number of experts per layer in HILO is calculated as $\text{Rank}_i/8*\text{AlphaLoRA}_i$, where Rank_i represents the rank value of adapters in layer i, and AlphaLoRA_i represents the number of adapter experts in layer i of AlphaLoRA, denoted as '‡‡' in this paper. Although HILO adopts the Top-2 activation scheme, the decreasing ranks in the shallow layers result in proportionally fewer active parameters. Consequently, the active parameters of HILO are reduced from 2, as in Vanilla, MoLA, and AlphaLoRA, to 1.25, representing a 37.50% reduction.

In terms of accuracy, since Llama 2 is not fine-tuned on new datasets, it demonstrates relatively low performance. The Vanilla attains higher accuracy at the expense of increasing the size of trainable and active parameters. However, HILO, with a smaller parameter size, still outperforms the Vanilla method on two datasets. Comparing MoLA and AlphaLoRA, as shown in Table 1, HILO is lower (0.08%↓) than MoLA only on one of 6 datasets (CQA) while achieving a maximum improvement of 2.00% on OQA. Moreover, HILO outperforms AlphaLoRA across all datasets, with the highest improvement of 1.97% on the MRPC and an average improvement of 1.01%.

Method	Adapter Expert Setting			Parameter ↓		Accuracy (%) ↑						
	Exp.	Rank	Top-K	Active	Trainable	SQA	CQA	OQA	MRPC	COLA	RTE	Avg.
Llama 2	-	-	-	-	-	40.23	20.58	25.20	66.55	55.99	53.42	43.66
Vanilla	8	8	2	2	1	90.10	78.21	76.20	85.15	84.85	87.72	83.71
MoLA	▽	8	2	2	0.63	89.61	**76.00**	74.20	84.23	85.52	84.47	82.34
HiLo	8	▽	2	**1.25**	0.63	**90.42**	75.92	**76.20**	**84.52**	**86.19**	**84.48**	**82.96**
AlphaLoRA	‡	8	2	2	0.63	89.70	75.83	74.60	82.49	85.13	85.55	82.22
HiLo	‡‡	▽	2	**1.25**	0.63	**90.15**	**76.58**	**75.80**	**84.46**	**84.27**	**88.09**	**83.23**

Table 1: Performances of HiLo (ours) and comparison methods

	Base	Vanilla	MoLA	HiLo	AlphaLoRA	HiLo
Llama 3	55.83	84.20	85.17	**86.35**	84.53	**86.12**
Qwen 2	59.01	78.94	78.44	**80.71**	78.73	**80.28**

Table 2: Average accuracies (%) of more base models. The detailed results are presented in Appendix D.

As shown in Table 2, under the same configuration, HiLo achieves higher average accuracy on both Llama 3 and Qwen 2 models. Specifically, while maintaining the same number of trainable parameters and reducing activation parameters by 37.50%, it outperforms MoLA and AlphaLoRA by 1.4% and 0.6% on Llama 3, and by 2.34% and 1.34% on Qwen 2, respectively.

The experimental results demonstrate that HiLo achieves higher model accuracy compared to MoLA and AlphaLoRA while introducing the same size of trainable parameters and a lower size of active parameters.

(2) HiLo *vs.* AdaMoE. To compare with AdaMoE, HiLo assigns the rank values from shallow to deep layers, grouped every 8 layers, to 2, 4, 6, and 8, while keeping the number of experts constant. In contrast, the activations of real and placeholder adapter experts are learned during model fine-tuning. Similarly, the trainable parameter size of AdaMoE is 1, whereas HiLo's is 0.63.

Additionally, we monitored the expert activation during validation. As shown in Table 3, the average active parameter of AdaMoE across datasets is 1.04, while HiLo's is 0.65, representing a 37.5% reduction. For accuracy, HiLo outperforms AdaMoE across all datasets, with the highest improvement of 3.26% on the RTE dataset and an average improvement of 1.04%.

The experimental results exhibit that HiLo achieves higher model accuracy with fewer trainable and active parameters compared to AdaMoE. HiLo also performs better on Llama 3 and Qwen 2 models, the results are shown in Appendix D.

Dataset	Active Parameter ↓		Accuracy (%) ↑	
	AdaMoE	HiLo	AdaMoE	HiLo
SQA	1.05	**0.68**	89.43	**90.06**
CQA	1.01	**0.63**	75.26	**75.43**
OQA	1.03	**0.63**	72.60	**73.60**
MRPC	1.03	**0.64**	84.81	**85.62**
COLA	1.01	**0.63**	85.42	**85.81**
RTE	1.09	**0.67**	81.94	**85.20**
Avg.	1.04	**0.65**	81.58	**82.62**

Table 3: **Performances of AdaMoE and HiLo with the base model of Llama 2**. Allocating 8 experts and 8 placeholder experts to each layer and employing a Top-2 activation strategy. AdaMoE's adapter rank is 8, and HiLo sets the rank to ▽ across layers.

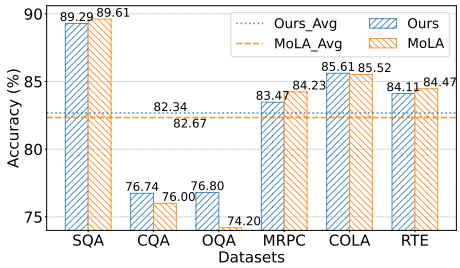


Figure 4: **Accuracies of Ours and MoLA**. Ours assigns experts and ranks as ▽ across layers.

5.2.2 Effect of Hierarchical Rank

In this experiment, the budget of trainable parameters is not kept consistent. HiLo reduces the rank as ▽ while maintaining the same number of experts as MoLA. With this setup, the total size of trainable parameter in HiLo is reduced to 0.39, indicating a 37.50% decrease compared to MoLA. Similarly, the size of active parameters in HiLo is reduced to 1.25, showing a 37.50% reduction.

As shown in Figure 4, HiLo outperforms MoLA on CQA, OQA, and COLA datasets and achieves higher average accuracy across all datasets. These results further demonstrate that HiLo can achieve higher fine-tuning accuracy with fewer trainable and active parameters compared to MoLA.

We integrate MoLA and AdaMoE as a new method, denoted by Mix, which combines layer-

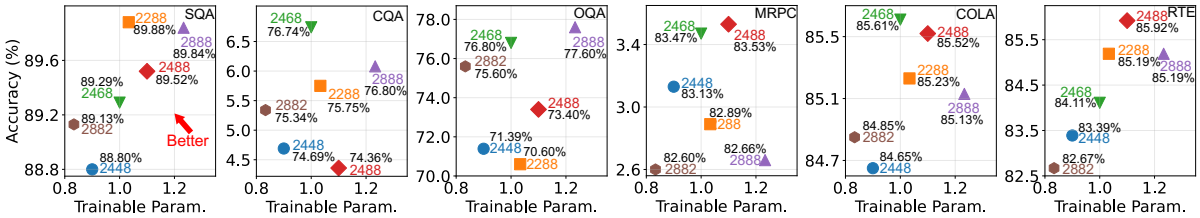


Figure 5: Performances of different rank configurations.

Adapter Experts		Parameter ↓		Accuracy (%) ↑
Rank	Top-K	Active	Trainable	
8	2	1.03	0.63	88.44
▽	2	**0.77**	**0.39**	**89.74**
8	▽	1.65	0.63	**90.56**
▽	▽	**1.23**	**0.39**	90.24

Table 4: **Performances of Mix and HiLo in SQA**. Mix and HiLo allocate experts and placeholder experts as ▽.

wise expert allocation and learning-based adapter expert activation. On this basis, we introduce HiLo as a comparative method based on Mix, with experiments conducted on the SQA dataset. Under the condition that trainable parameters are unaffected by the activation scheme, the trainable parameters of Mix and HiLo are 0.63 and 0.39, respectively, with HiLo achieving a 37.50% reduction.

According to statistical analysis, with the Top-2 activation scheme, the active parameters of Mix and HiLo are 1.03 and 0.77, respectively, showing a 25.24% reduction. Although the Top-▽ activation is not practically applied, we conducted validation and evaluation under this setting, where Mix and HiLo exhibit active parameters of 1.65 and 1.23, respectively, presenting a 25.45% reduction.

In terms of accuracy, as shown in Table 4, under the de facto widely employed Top-2 activation scheme, HiLo outperforms Mix by 1.30%. Under the Top-▽, HiLo 's accuracy is 0.32% lower than Mix, but it achieves higher accuracy than Vanilla, MoLA, and AlphaLoRA, while reducing both trainable and active parameters simultaneously.

5.2.3 Robustness and Sensitivity Exploration

To evaluate the effectiveness of the proposed rank configuration, we compare multiple layer-wise rank patterns under comparable parameter budgets. The results consistently show that the monotonic linear configuration (e.g., 2468) achieves the best trade-off between accuracy and parameter efficiency.

We further evaluate the performance of all ranks are set to 2 and 8, the min and max values of 2468, both of which correspond to a grouping granularity of l=32. Compared to the rank of 2468, the 2222-rank reduces parameter size but leads to an average accuracy drop of 1.93% and 1.88% under experts of 8888 and 2468, respectively. Meanwhile, the 8888-rank requires larger parameter footprint but only yields a marginal gain of 0.75% for 8888-experts, and even results in a 0.33% decline for 2468-experts. These results indicate that extreme configurations (*e.g.*, uniformly low or high ranks) are suboptimal, while the linear schedule provides a more balanced performance across layers. The detail results are presented in Appendix E.

5.2.4 Computing and Resource Overhead

To evaluate the computing and resource efficiency, we profiled HiLo and MoLA following the experimental setup described in §5.2.2. To present the results more clearly, the computational and resource costs reported exclude the contributions from the base model. Compared with MoLA, HiLo reduces the tuning-related memory footprint by 31.84%, lowering the overhead from 1632 to 1112 MB and thereby decreasing peak training memory usage by approximately 520 MB. In addition, HiLo reduces the per-step forward computation by 3.4 GFLOPs (from 1282.2 to 1278.8 GFLOPs, including the base model's fixed 1274.1 GFLOPs), corresponding to a 41.64% reduction in fine-tuning computation cost. Furthermore, HiLo decreases forward latency by 8 ms and backward propagation latency by 25.8 ms.

Based on the above results, HiLo introduce none additional resource overhead compared with existing approaches, verifying HiLo achieves simultaneous improvements in both accuracy and efficiency.

6 Conclusion

We propose HiLo, a training-free configuration for mixture of adapter experts that jointly allocates expert number and rank. By introducing hierarchical rank assignment, HiLo improves parameter utilization and achieves a better accuracy–efficiency trade-off. Experiments show consistent gains over existing methods while reducing both trainable and active parameters, making it suitable for practical deployment under limited tuning budgets.

Limitations

HILO employs a static linear allocation to maintain its training-free and ease of implementation across different models. While our experiments demonstrate that this heuristic is robust and effective, we recognize that it does not account for potential benefits from dynamic or adaptive rank adjustment during the training process. Future research could explore adaptive mechanisms that further refine these configurations, building upon the foundational principles established in this work.

Ethical considerations

The proposed HILO method is a technical framework designed to enhance the computational efficiency of fine-tuning large language models. It does not involve the collection or use of any private or sensitive data. All experiments in this paper are conducted using publicly available models and datasets for academic research purposes. The deployment of our method does not inherently violate any ethical principles or social norms.

References

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, and 1 others. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, and 1 others. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In *The 62nd Annual Meeting of the Association for Computational Linguistics*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, and 1 others. 2024. Loramoe: Alleviating world knowledge forgetting in large language models via moe-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1932–1945.

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

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*.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. Lora+: Efficient low rank adaptation of large models. In *Forty-first International Conference on Machine Learning*.

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, 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. In *The Tenth International Conference on Learning Representations*.

Quzhe Huang, Zhenwei An, Nan Zhuang, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. 2024. Harder tasks need more experts: Dynamic routing in moe models. In *The 62nd Annual Meeting of the Association for Computational Linguistics*.

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

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M Asano. 2024. Vera: Vector-based random matrix adaptation. In *The Twelfth International Conference on Learning Representations*.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. Gshard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*.

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

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

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.

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

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2024b. Gpt understands, too. *AI Open*, 5:208–215.

Pan Lu, Swaroop Mishra, Tanglin 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. *Advances in Neural Information Processing Systems*, 35:2507–2521.

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.

Basil Mustafa, Carlos Riquelme, Joan Puigcerver, Rodolphe Jenatton, and Neil Houlsby. 2022. Multimodal contrastive learning with limoe: the language-image mixture of experts. *Advances in Neural Information Processing Systems*, 35:9564–9576.

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.

Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, and 1 others. 2021. Hash layers for large sparse models. *Advances in Neural Information Processing Systems*, 34:17555–17566.

Steffen Schotthöfer, Timon Klein, and Jonas Kusch. 2025. A geometric framework for momentum-based optimizers for low-rank training. *arXiv preprint arXiv:2506.17475*.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

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*.

Haoyuan Wu, Haisheng Zheng, Zhuolun He, and Bei Yu. 2024a. Parameter-efficient sparsity crafting from dense to mixture-of-experts for instruction tuning on general tasks. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 737–749.

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

Shu Yang, Muhammad Asif Ali, Cheng-Long Wang, Lijie Hu, and Di Wang. 2024a. Moral: Moe augmented lora for llms' lifelong learning. *arXiv preprint arXiv:2402.11260*.

Yuanhang Yang, Shiyi Qi, Wenchao Gu, Chaozheng Wang, Cuiyun Gao, and Zenglin Xu. 2024b. Xmoe: Sparse models with fine-grained and adaptive expert selection. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 11664–11674.

Zihao Zeng, Yibo Miao, Hongcheng Gao, Hao Zhang, and Zhijie Deng. 2024. Adamoe: Token-adaptive routing with null experts for mixture-of-experts language models. In *Findings of the Association for Computational Linguistics ACL 2024*.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2025. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*.

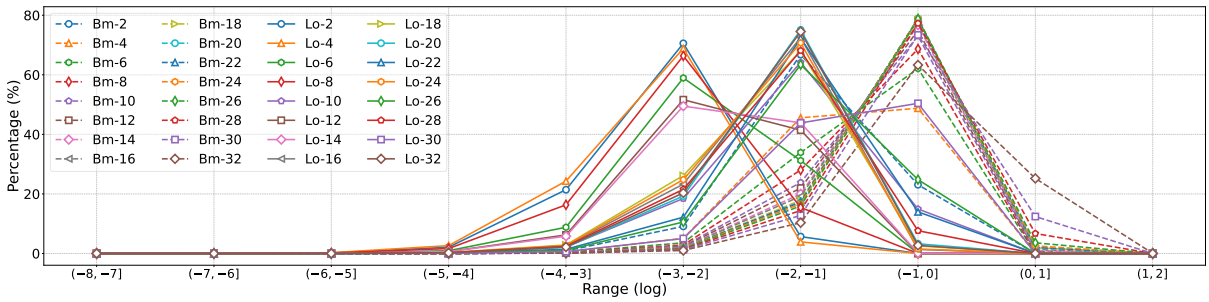
Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, and 1 others. 2022. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114.

Appendix

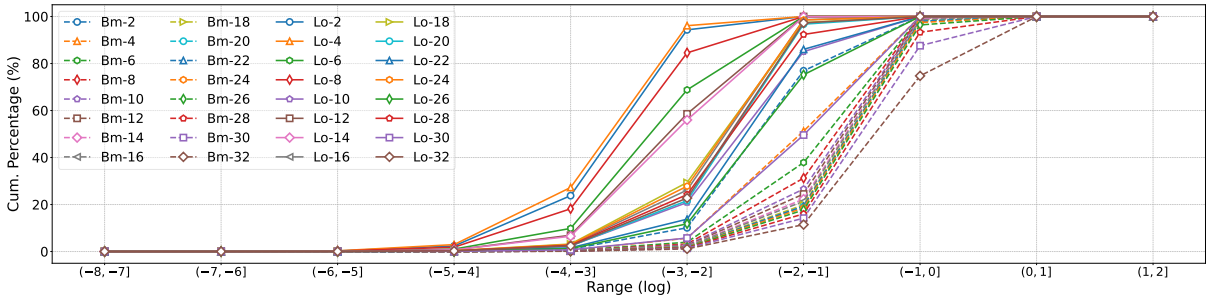
A Output Analysis

A.1 Layer-wise Output Magnitudes of Adapter Experts

To further examine whether adapter behavior varies across layers, we analyze the absolute output magnitudes of the base model feed-forward networks



(a) Distribution of output values



(b) Cumulative distribution of output values

Figure A: **Comparison of output value distributions**: Base model (Bm-layer$_i$) *vs.* LoRA (Lo-layer$_i$).

and the corresponding LoRA adapter experts. The analysis is conducted under the Top-2 activation setting. For each layer, we collect the absolute output values of the frozen feed-forward module and the activated LoRA adapters, and compare their layer-wise distributions.

As shown in Figure A, the output magnitudes of the frozen feed-forward networks are consistently larger than those of the LoRA adapters, typically by one to two orders of magnitude. This is expected, since the frozen backbone performs the main computation, while LoRA adapters introduce lightweight task-specific adjustments. Despite their smaller scale, adapter outputs still provide useful evidence for understanding how fine-tuning signals vary across layers.

The LoRA adapter outputs exhibit clear layer-wise variation. In particular, shallow layers generally have smaller adapter output magnitudes than deeper layers, suggesting that adapter-induced adjustments are not uniformly distributed across depth. This observation is consistent with the Frobenius-norm analysis in the main text, where adapter update magnitudes also vary across layers. These results further support our motivation to treat adapter rank as a configurable factor in MoAE, rather than fixing it uniformly for all layers.

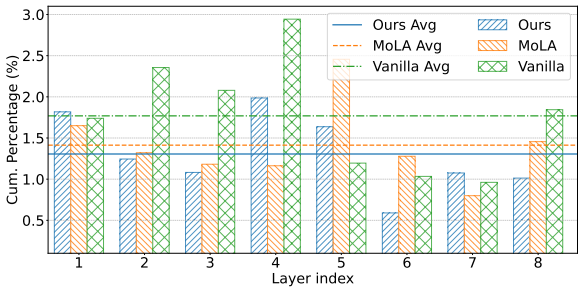


Figure B: Proportions of values (<10^{-3}) of Vanilla method, MoLA, and HiLo.

A.2 Comparisons of Shallow Layer Outputs

We also analyzed the proportion of values smaller than the peak range (*i.e.*, less than 10^{-3}) in the shallow layers (from 1st to 8th layers) for Vanilla, MoLA, and HiLo. As shown in Figure B, the detailed results are as follows. (1) The proportions of the three methods in the 1st layer are very close; (2) Except for the 1st layer, MoLA has lower proportions than Vanilla and HiLo in 4 and 2 layers, respectively; (3) HiLo exhibits lower proportions than Vanilla and MoLA in five layers. Overall, HiLo indeed strengthens the fine-tuning signals of adapters in the shallow layers and, as expected, achieves higher overall accuracy than the comparison methods.

Method	Adapter Expert Setting			Parameter ↓		Accuracy (%) ↑						
	Exp.	Rank	Top-K	Active	Trainable	SQA	CQA	OQA	MRPC	COLA	RTE	Avg.
Llama 3	-	-	-	-	-	83.73	43.08	31.40	58.43	49.37	68.95	55.83
Vanilla	8	8	2	2	1	94.78	70.03	85.40	83.65	84.18	87.36	84.23
MoLA	▽	8	2	2	0.63	94.24	78.54	84.80	83.13	84.37	85.92	85.17
HiLo	8	▽	2	**1.25**	0.63	**94.83**	**80.75**	**86.40**	**83.77**	**85.33**	**87.00**	**86.35**
AlphaLoRA	‡	8	2	2	0.63	94.37	78.86	82.60	81.44	84.94	85.55	84.63
HiLo	‡‡	▽	2	**1.25**	0.63	**94.56**	**79.85**	**84.40**	**85.22**	**85.13**	**88.09**	**86.21**

Table A: Performances of HiLo and comparison methods with base model Llama 3

Method	Adapter Expert Setting			Parameter ↓		Accuracy (%) ↑						
	Exp.	Rank	Top-K	Active	Trainable	SQA	CQA	OQA	MRPC	COLA	RTE	Avg.
Qwen 2	-	-	-	-	-	62.86	21.62	52.80	73.39	69.41	74.00	59.01
Vanilla	8	8	2	2	1	87.00	70.27	74.00	76.57	82.74	83.03	78.94
MoLA	▽	8	2	2	0.63	84.89	70.68	72.60	78.02	81.78	82.67	78.44
HiLo	8	▽	2	**1.25**	0.63	**87.23**	**71.33**	**78.00**	**81.28**	**83.41**	**83.03**	**80.71**
AlphaLoRA	‡	8	2	2	0.63	87.14	70.43	74.80	77.15	81.97	80.87	78.73
HiLo	‡‡	▽	2	**1.25**	0.63	**88.48**	**72.40**	**76.60**	**78.67**	**83.22**	**82.31**	**80.28**

Table B: Performances of HiLo and comparison methods with base model Qwen 2

Dataset	Active Parameter ↓		Accuracy (%) ↑	
	AdaMoE	HiLo	AdaMoE	HiLo
SQA	1.07	**0.67**	91.05	**92.63**
CQA	1.04	**0.65**	77.97	**80.09**
OQA	1.08	**0.68**	86.40	**86.40**
MRPC	1.13	**0.70**	83.53	**84.64**
COLA	1.12	**0.69**	84.85	**85.43**
RTE	1.09	**0.68**	87.00	**87.72**
Avg.	1.09	**0.68**	85.13	**86.15**

Table C: **Performances of AdaMoE and HiLo with base model Llama 3**. Configuration is consistent with Table 3.

Dataset	Active Parameter ↓		Accuracy (%) ↑	
	AdaMoE	HiLo	AdaMoE	HiLo
SQA	1.08	**0.63**	87.09	**87.46**
CQA	1.05	**0.65**	70.76	**71.58**
OQA	1.16	**0.71**	77.20	**78.60**
MRPC	1.16	**0.73**	80.23	**80.87**
COLA	1.05	**0.66**	81.87	**82.07**
RTE	1.06	**0.70**	80.50	**81.95**
Avg.	1.09	**0.68**	79.61	**80.42**

Table D: **Performances of AdaMoE and HiLo with base model Qwen 2**. Configuration is consistent with Table 3.

B Detailed Description of Baselines

This section provides a detailed description of the baseline methods used in our experiments.

MoLA (Gao et al., 2024): This method follows a manual, layer-wise allocation strategy. It allocates fewer experts to shallow layers and more to deeper layers, based on the observation that deeper layers require more fine-tuning capacity. For the model, the optimal allocation with a fixed parameter budget is to group every 8 layers and assign 2, 4, 6, and 8 experts, respectively, from shallow to deep layers. In this paper, this allocation strategy is denoted as ▽.

AlphaLoRA (Qing et al., 2024): This method takes a data-driven approach by allocating experts based on the training quality of the original model layers. It uses the Heavy-Tailed Self-Regularization theory to evaluate each layer's quality and assigns a variable number of experts accordingly. For the Llama 2-7B model, the layer-wise allocation strategy is as follows: 1, 3, 4, 4, 4, 4, 3, 3, 3, 3, 2, 2, 3, 3, 3, 4, 5, 5, 7, 6, 8, 9, 8, 7, 8, 6, 6, 8,

Adapter Experts	8888			2468 (▽)		
Rank	2468	2222	8888	2468	2222	8888
Trainable Param.	1	0.25	1.6	0.75	0.25	1
SQA	90.42	$88.03_{\downarrow 2.39}$	$90.10_{\downarrow 0.32}$	89.29	$88.03_{\downarrow 1.26}$	$89.61_{\uparrow 0.32}$
CQA	75.92	$73.79_{\downarrow 2.13}$	$78.21_{\uparrow 2.29}$	76.74	$73.79_{\downarrow 2.95}$	$76.00_{\downarrow 0.74}$
OQA	76.20	$75.20_{\downarrow 1.00}$	$76.20_{-0.00}$	76.80	$75.20_{\downarrow 1.60}$	$74.20_{\downarrow 2.60}$
MRPC	84.52	$82.20_{\downarrow 2.32}$	$85.15_{\uparrow 0.63}$	83.47	$82.20_{\downarrow 1.27}$	$84.23_{\uparrow 0.76}$
COLA	86.19	$84.65_{\downarrow 1.54}$	$84.85_{\downarrow 1.34}$	85.61	$84.65_{\downarrow 0.96}$	$85.52_{\downarrow 0.09}$
RTE	84.48	$82.31_{\downarrow 2.17}$	$87.72_{\uparrow 3.24}$	84.11	$80.86_{\downarrow 3.25}$	$84.47_{\uparrow 0.36}$
Avg.	82.96	$81.03_{\downarrow 1.93}$	$83.71_{\uparrow 0.75}$	82.67	$80.79_{\downarrow 1.88}$	$82.34_{\downarrow 0.33}$

Table E: Performances of configuration rank 2222 and 8888

7, 7, 9, and 5. In this paper, this strategy is denoted as ‡.

The above two methods focus on Expert Allocation, which optimizes the number of adapter experts E_i per layer i to achieve better performance under a constrained parameter budget.

AdaMoE (Zeng et al., 2024): This method improves efficiency by dynamically adjusting the number of activated experts. It achieves this by introducing a set of "placeholder" experts that have no parameters and do not perform computation. During training, the gating network learns to activate a variable number of real experts by also selecting placeholder experts. In our experiments, we set a 1:1 ratio of real experts to placeholder experts for each layer and employ a Top-2 activation policy for a fair comparison. This means that for each token, the gating network selects 2 experts in total, which can be any combination of 0, 1, or 2 real experts and the corresponding number of placeholder experts.

The above method focuses on *Expert Activation*, which optimizes the number of activated experts per token to reduce computational cost during inference.

C Detail results of §5.2.3

The detailed results for various rank configurations under different adapter experts are presented in Table E. We define the trainable parameter size of the configuration with 8888 adapter experts and ranks of 2468 (*i.e.*, HILO) as the unit of 1. Within these settings, the 2468 adapter experts with 8888 ranks corresponds to MoLA, while the 2468 adapter experts with 2468 ranks represents the Mix setup described in §5.2.2. Changes in relative accuracy are computed against the 2468 rank configuration within the corresponding adapter experts group.

D Detailed results of Llama 3 and Qwen 2

The accuracy results with Llama 3 and Qwen 2 as base models are shown in Table A to D. The results show that HILO also achieves better performance on Llama 3 and Qwen 2 compared to the existing methods. It can also indicate that HILO has good generalization, which is not limited to the specified model.

E Performance of all ranks set to 2 or 8

In the experiments conducted in this work, the rank values included 2, 4, 6, and 8. The results when all rank values were set to the minimum or maximum value (2 or 8) are shown in Table E.