AlphaLoRA: Assigning LoRA Experts Based on Layer Training Quality

Peijun Qing[1] Chongyang Gao[2] Yefan Zhou[1] Xingjian Diao[1]
Yaoqing Yang[1] Soroush Vosoughi[1†]
[1]Dartmouth College [2]Northwestern University
{peijun.qing.gr, yefan.zhou.gr, yaoqing.yang, soroush.vosoughi}@dartmouth.edu
chongyanggao2026@u.northwestern.edu

Abstract

Parameter-efficient fine-tuning methods, such as Low-Rank Adaptation (LoRA), are known to enhance training efficiency in Large Language Models (LLMs). Due to the limited parameters of LoRA, recent studies seek to combine LoRA with Mixture-of-Experts (MoE) to boost performance across various tasks. However, inspired by the observed redundancy in traditional MoE structures, prior studies find that LoRA experts within the MoE architecture also exhibit redundancy, suggesting a need to vary the allocation of LoRA experts across different layers. In this paper, we leverage Heavy-Tailed Self-Regularization (HT-SR) Theory to design a fine-grained allocation strategy. Our analysis reveals that the number of experts per layer correlates with layer training quality, which exhibits significant variability across layers. Based on this, we introduce AlphaLoRA, a theoretically principled and training-free method for allocating LoRA experts to reduce redundancy further. Experiments on three models across ten language processing and reasoning benchmarks demonstrate that AlphaLoRA achieves comparable or superior performance over all baselines. Our code is available at https://github.com/morelife2017/alphalora.

1 Introduction

LLMs have shown impressive performance on various NLP tasks (Brown et al., 2020; Chowdhery et al., 2022; Touvron et al., 2023b; Jiang et al., 2023; Jian et al., 2024; Zhang et al., 2024). However, due to the increasing size of modern LLMs, significant computational resources are required for full fine-tuning. To address this issue, researchers are increasingly focusing on PEFT methods to reduce training costs, such as Adapter-tuning (Houlsby et al., 2019) and LoRA (Hu et al., 2022). Despite their training efficiency, the performance of PEFT methods in fine-tuning LLMs is still limited due to the small number of parameters (Xu et al., 2023).

To address the limitation, recent studies seek to combine LoRA and MoE by adding multiple LoRA modules (Liu et al., 2023; Gao et al., 2024; Luo et al., 2024). The MoE structure in LLMs typically consists of multiple feed-forward "sub-networks", or "experts", that are trained to handle different types of inputs or tasks (Shazeer et al., 2017a). MoE structures are designed to dynamically activate only a subset of experts for each input, significantly scaling up the number of parameters, while incurring an affordable computational overhead.

Existing LoRA-MoE methods integrate multiple LoRA modules into each sublayer of the transformer block and employ different strategies to assign experts to different tokens or tasks (Li et al., 2024; Huang et al., 2023; Zhang et al., 2023; Yang et al., 2024; Dou et al., 2024; Luo et al., 2024; Feng et al., 2024; Liu et al., 2023). For example, at the task level, Huang et al. (2023); Zhang et al. (2023) explore various composition strategies to combine multiple LoRA experts trained individually on different tasks. At the token level, both experts and input tokens interact with a routing network, resulting in the activation of experts based on the characteristics of the input tokens (Gao et al., 2024; Dou et al., 2023). These methods demonstrate superior performance in single-task and multi-task learning compared to standard LoRA, with the number of experts per layer uniformly distributed, such as three per layer, as shown in Table 1.

However, recent studies (Chen et al., 2023; Zoph et al., 2022) in MoE show that employing a large number of experts may be redundant due to representational collapse or learned routing policy overfitting. Similarly, Gao et al. (2024) investigate redundancy in parameter-efficient MoE. Unlike existing LoRA-MoE methods that uniformly allocate experts across all layers, they manually design four architectures with varying group-wise expert allo-

[†]Corresponding author.

cations. Specifically, they divide the 32 layers of the LLaMA-2 model (Touvron et al., 2023b) into 4 groups, with the first 8 layers constituting the first group, and so forth. They allocate a varying number of experts to each group (layers within the same group have the same number of experts). While their allocation strategy provides insights into overall architecture design, suggesting that higher layers need more LoRA experts, research on achieving more effective integration remains in its early stages.

To create a theoretically sound allocation strategy aimed at minimizing expert redundancy, our research draws inspiration from the Heavy-Tailed Self-Regularization (HT-SR) Theory (Martin and Mahoney, 2019, 2020, 2021; Martin et al., 2021). The HT-SR theory examines the properties of heavy-tailed (HT) structures observed in the Empirical Spectral Density (ESD) of weight matrices. The application of HT-SR to model selection and layer-wise adaptive training (Zhou et al., 2024; Yang et al., 2023) showcases the theory's effectiveness in assessing both model and layer quality.

In this paper, we propose a fine-grained strategy for allocating layer-wise expert numbers, namely, `AlphaLoRA`. According to Zhou et al. (2024), layers with more pronounced HT properties are generally well-trained. Following Zhou et al. (2024), we measure the HT characteristics by fitting a power law (PL) distribution to the ESD and use the exponent as the metric to measure HT properties. We then use the Hill estimator (Hill, 1975; Zhou et al., 2024) to calculate `PL_Alpha_Hill`. The core idea behind `AlphaLoRA` is to allocate fewer experts to better-trained (more HT) layers, which is indicated by lower `PL_Alpha_Hill` values, thus reducing the redundancy in a more theoretically-principled manner. The contributions of our work are summarized as follows:

- We are the first to interpret the correlation between layer-wise training quality and LoRA expert number through the lens of HT-SR theory. Empirical results on three widely-used language models suggest that well-trained layers need fewer LoRA experts.

- We propose a fine-grained allocation strategy, `AlphaLoRA`, for allocating layer-wise expert numbers. Inspired by HT-SR theory, this method is theoretically grounded and training-free. `AlphaLoRA` generally outperforms MoLA-$\triangledown$, the current state-of-the-art non-uniform expert allocation method, across three models and ten NLP datasets. Notably, `AlphaLoRA` with 80 experts surpasses MoLA-$\triangledown$(2468) with 160 experts, achieving comparable performance with 50% fewer parameters.

- We compare several layer-wise weight matrix metrics from HT-SR theory for evaluating layer training quality and allocating expert numbers. The relative performance of these metrics reveals that the `PL_Alpha_Hill` metric outperforms others, corroborating the findings of (Zhou et al., 2024) that the `PL_Alpha_Hill` metric is better for assessing layer training quality. This further demonstrates the correlation between layer expert number and layer training quality.

Granularity	Method
Uniform	MixLoRA(Li et al., 2024) MoRAL (Yang et al., 2024) LoRAMoE (Dou et al., 2024) MoELoRA (Luo et al., 2024) MoA (Feng et al., 2024) MOELoRA (Liu et al., 2023)
Group-wise	MoLA (Gao et al., 2024)
Layer-wise	**`AlphaLoRA`**

Table 1: Comparison of allocation strategy between different LoRA-MoE methods.

2 Preliminary

2.1 LoRA-MoE Architecture

The LoRA-MoE architecture creates multiple LoRA experts for each layer in a pre-trained LLM. The "LoRA expert" used in this work refers to the vanilla LoRA block (Hu et al., 2022). For a pre-trained weight matrix $\mathbf{W_0} \in \mathbb{R}^{m \times n}$ $(n < m)$, LoRA creates two low-rank trainable matrices $\mathbf{A}$ and $\mathbf{B}$, where $\mathbf{A} \in \mathbb{R}^{m \times r}$, $\mathbf{B} \in \mathbb{R}^{r \times n}$, and $r \ll \min(m, n)$. Thus, the dimension of $\mathbf{ABx}$ equals the dimension of $\mathbf{W_0x}$ for the input $\mathbf{x}$. During training, $\mathbf{W_0}$ is frozen while $\mathbf{A}$ and $\mathbf{B}$ receive gradient updates. The output $\mathbf{h}$ is expressed as follows:

$$\mathbf{h} = \mathbf{W}_0\mathbf{x} + \Delta\mathbf{W}\mathbf{x} = \mathbf{W}_0\mathbf{x} + \mathbf{ABx}. \quad (1)$$

Each LoRA-MoE layer contains N LoRA experts, which is denoted as $\{\sum_{i=1}^{N}\}$. The forward process

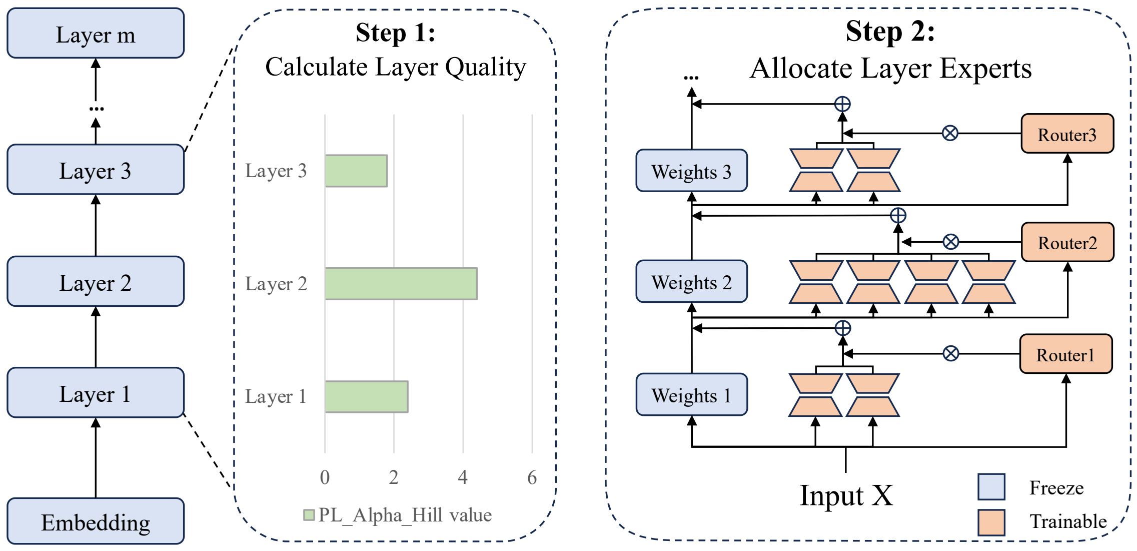

Figure 1: The overview of AlphaLoRA. For a transformer-based model with m layers, AlphaLoRA involves two steps: (Step 1) Conducting ESD analysis on each layer and applying PL fitting to obtain the layer-wise PL_Alpha_Hill value. (Step 2) Converting the layer-wise PL_Alpha_Hill value into the number of experts using a mapping function, followed by initializing the experts for each layer. For instance, Weights 1 represents all the weight matrices (such as attention weight matrix and projection weight matrix) in layer 1.

of the layer is expressed as:

$$\mathbf{o} = \mathbf{W}_0\mathbf{x} + \Delta\mathbf{W}\mathbf{x} = \mathbf{W}_0\mathbf{x} + \sum_{i=1}^{N} \mathbf{G}(\mathbf{x})_i \mathbf{E}_i(\mathbf{x}), \tag{2}$$

where $\mathbf{E}_i(\mathbf{x})$ and $\mathbf{G}(\mathbf{x}) = \text{Softmax}(\mathbf{x}\mathbf{W}_g)$ represent the i-th expert and the router in the LoRA-MoE layer, respectively. The $\mathbf{W}_g$ is the trainable parameter matrix of the route network used to allocate input x to different experts. The router enables experts to develop varied capabilities and efficiently handle various types of tasks and inputs.

2.2 ESD Shape Metric

Empirical Spectral Density (ESD). ESD represents the distribution of eigenvalues of a matrix, often used to understand properties of large random matrices that arise in various applications such as neural networks (Martin and Mahoney, 2021). Let $\mathbf{A} = \mathbf{W}_0^\top \mathbf{W}_0 \in \mathbb{R}^{n\times n}$ be a symmetric matrix with eigenvalues $\{\lambda_i^{\mathbf{A}}\}_{i=1}^n$. The empirical spectral density (ESD) of $\mathbf{A}$ is defined as the probability measure:

$$\rho(\lambda; \mathbf{A}) = \frac{1}{n}\sum_{i=1}^{n} \delta\left(\lambda - \lambda_i^{\mathbf{A}}\right), \tag{3}$$

where δ is the Dirac delta function. The ESD represents the distribution of the eigenvalues of $\mathbf{A}$.

Heavy-Tailed Self-Regularization (HT-SR) Theory. Drawing from Random Matrix Theory (Tao, 2023; Bai and Silverstein, 2010; Couillet and Liao, 2022), HT-SR theory relies on the empirical observation that well-trained models usually display strong correlations, leading to HT structures in the ESD of each layer (Martin et al., 2021; Yang et al., 2023; Zhou et al., 2024). Derived from HT-SR theory, *shape metrics* are analytical methods used to characterize the HT properties of ESDs in neural networks, which correlate with the shapes emerging in their ESDs. In this work, we mainly study three shape metrics: PL_Alpha_Hill, Alpha_Hat, and Stable_Rank. Inspired by previous work on estimating model quality (Zhou et al., 2024; Yang et al., 2023), we apply PL_Alpha_Hill, which is proved to be the most effective, as the main metric to evaluate layer quality. The definition of PL_Alpha_Hill is explained in §3.1. Other definitions can be found in the Appendix A.

3 Method

In this section, we introduce AlphaLoRA, an expert allocation strategy based on the PL_Alpha_Hill metric. Given a transformer-based model, we first compute the PL_Alpha_Hill metric value for each layer. §3.1 elaborates on the calculation of layer-wise PL_Alpha_Hill metric. §3.2 gives the process of mapping the layer-wise PL_Alpha_Hill value to expert number. The overview of our method is illustrated in Figure 1.

3.1 Analyzing Layer Training Quality

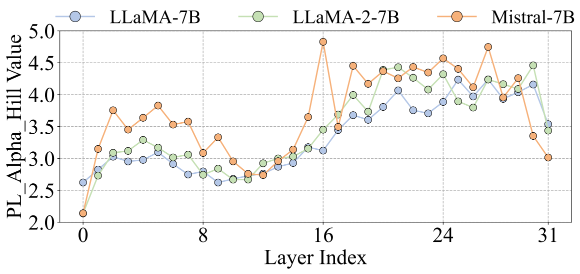

Figure 2: Illustration of the PL_Alpha_Hill values for each layer across three different models.

AlphaLoRA measures the layer training quality based on the HT characteristic of the layer ESDs, which is quantified by the HT metric PL_Alpha_Hill. For the i-th weight matrix in each layer from a transformer-based model, we first calculate the eigenvalues of its correlation matrix $\mathbf{X}_i = \mathbf{W}_i^T \mathbf{W}_i$ and ESD ρ. We then fit a power law (PL) distribution p to the HT part of the ESD, taking values within an interval $(\lambda_{\text{min}}, \lambda_{\text{max}})$, which is defined as:

$$p(\lambda) \propto \lambda^{-\alpha}, \quad \lambda_{\text{min}} < \lambda < \lambda_{\text{max}}. \tag{4}$$

We refer to its exponent α as PL_Alpha, where a lower value means more Heavy-tailed. We then use the Hill estimator (Hill, 1975) to calculate PL_Alpha_Hill as the following:

$$\texttt{PL\_Alpha\_Hill} = 1 + \frac{k}{\left(\sum_{i=1}^{k} \ln \frac{\lambda_{n-i+1}}{\lambda_{n-k}}\right)}, \tag{5}$$

where k is a parameter for controlling the lower eigenvalue threshold λ_{min} for PL estimation. We apply the Fix-finger method (Yang et al., 2023) to select the k, which keeps λ_{min} at the peak of the ESD. We calculate PL_Alpha_Hill for each weight matrix within a layer separately and compute the average to represent the layer value. Figure 2 shows the PL_Alpha_Hill values for three popular language models. The metric values show non-uniform distributions across layers, indicating varying training quality between layers.

Applications of HT-SR theory (Zhou et al., 2024) indicate assigning larger learning rates to undertrained layers helps these layers capture more correlations (or features) from the data (Wang et al., 2024). This implies that under-trained layers, which have captured fewer features, may need more LoRA experts to acquire additional features during fine-tuning.

3.2 Allocating Expert Based on Layer Quality

Given a Transformer model with m layers, we use s_i to denote the number of experts in layer i. We first calculate the PL_Alpha_Hill metric value for each layer using the algorithm in §3.1 to obtain a sequence of metric values $\mathcal{V} = [v_1, v_2, \ldots, v_m]$.

The number of experts assigned to each layer is determined using a mapping function $\psi : \mathbb{R}^m \rightarrow \mathbb{R}^m$. This function converts a sequence of metric values $\mathcal{V} = [v_1, v_2, \ldots, v_m]$ into the corresponding expert numbers $\mathcal{S} = [s_1, s_2, \ldots, s_m]$, represented as:

$$s_i = \left\lfloor \left(\frac{v_i^\beta}{\sum_{i=1}^m v_i^\beta} \right) \times T \right\rceil, \tag{6}$$

where $\lfloor\cdot\rceil$ denotes rounding to the nearest integer, and β is an exponent parameter that controls the standard deviation of the sequence $\mathcal{S}$. The values in $\mathcal{V}$ are normalized by dividing by their sum, ensuring that the total allocation across all layers is proportional to their relative importance. We introduce a target sum parameter T such that $\sum_{i=1}^m s_i = T$. Multiplying by T scales this proportional allocation to the desired number of experts, and finally, rounding to the nearest integer ensures discrete expert numbers for each layer. After rounding, if the total number of experts across all layers doesn't equal the target number T, we iteratively increment or decrement the integer value as follows:

$$\begin{cases} \text{if } \sum_{i=1}^m s_i < T, & s_{\arg\min(v_i^e - s_i)} + 1 \\ \text{if } \sum_{i=1}^m s_i > T, & s_{\arg\max(v_i^e - s_i)} - 1 \end{cases}, \tag{7}$$

where we identify a particular index in $\mathcal{S}$ that has the minimum (or maximum) difference between the scaled value and the present integer value, indicating the specific integer that needs to be incremented (or decremented) when the current sum is below (or above) the target sum. After getting the number of expert $\mathcal{S} = [s_1, s_2, \ldots, s_m]$ for each layer, we allocate s_i experts for layer i in a transformer-based model with m layers, resulting in $\sum_{i=1}^m s_i = T$ experts in total. Given a pre-trained weight matrix $\mathbf{W}_0^{i,t} \in \mathbb{R}^{m \times n}$ from module t in layer i, we create s_i pairs of low-rank matrices $\{\mathbf{A}_j^{i,t}, \mathbf{B}_j^{i,t}\}_{j=1}^{s_i}$. Each matrix $\mathbf{A}_j^{i,t}$ is initialized from a random Gaussian distribution, while $\mathbf{B}_j^{i,t}$ is set to zero, where

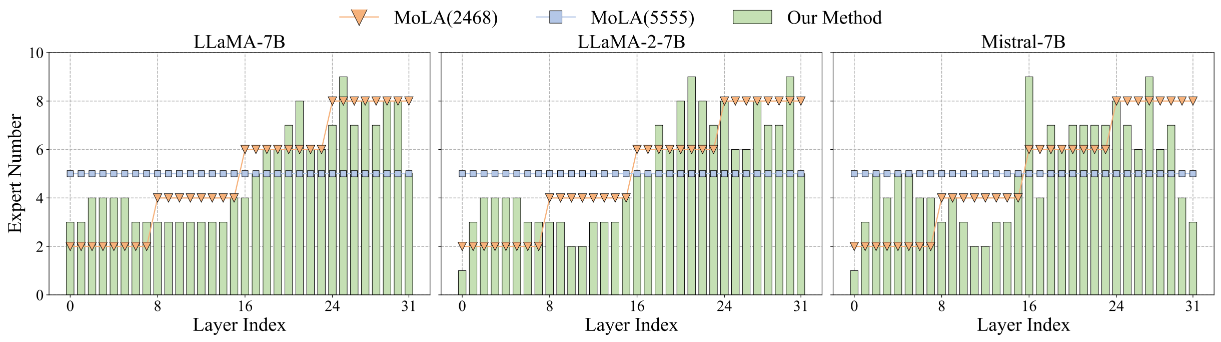

Figure 3: Comparing layer expert number assigned by `AlphaLoRA` and MoLA. MoLA(2468) allocates **2** experts to each layer for the first 8 layers, **4** experts to each layer for 9-16 layers, **6** experts to each layer for 17-24 layers, and **8** experts to each layer for the last 8 layers, which is denoted as **2468**. MoLA(5555) assigns a uniform **5** experts to each layer. The total number of experts is set at 160.

$\mathbf{A}_j^{i,t} \in \mathbb{R}^{m\times r}$, $\mathbf{B}_j^{i,t} \in \mathbb{R}^{r\times n}$, and $r \ll \min(m,n)$ (Hu et al., 2022).

A router $\mathbf{S}_j^{i,t}$ with a trainable weight matrix $\mathbf{W}_r^{i,t} \in \mathbb{R}^{n\times N_j}$ assigns experts for input $\mathbf{x}$. Following MoLA (Gao et al., 2024), we use the top-K strategy for computation and apply a load balancing loss at each layer (Zoph et al., 2022). This process is mathematically represented as follows:

$$\mathbf{S}_j^{i,t}(\mathbf{x}) = \frac{\text{TopK}(\text{Softmax}(\mathbf{W}_r^{i,t}\mathbf{x}), K)_j}{\sum_{j=1}^{K} \text{TopK}(\text{Softmax}(\mathbf{W}_r^{i,t}\mathbf{x}), K)_j}, \tag{8}$$

$$\mathbf{h}^{i,t} = \mathbf{W}_0^{i,t}\mathbf{x} + \sum_{j=1}^{K} \mathbf{S}_j^{i,t}(\mathbf{x})\mathbf{A}_j^{i,t}\mathbf{B}_j^{i,t}\mathbf{x}. \tag{9}$$

The output $\mathbf{h}^{i,t}$ is obtained by adding the transformation of $\mathbf{x}$ via the pre-trained weight matrix $\mathbf{W}_0^{i,t}$ to the aggregated low-rank transformations from the top K experts, where each expert's contribution is modulated by its corresponding assignment probability $\mathbf{S}_j^{i,t}(\mathbf{x})$.

4 Experiment

We design two experimental settings to examine the performance of `AlphaLoRA`, including direct fine-tuning and instruction-tuning→zero-shot evaluation. §4.1 compares the expert allocation between `AlphaLoRA` and variants of MoLA (Gao et al., 2024). §4.2 presents the results for two experimental settings. Implementation details can be found in Appendix C.

Models and datasets. To demonstrate the effectiveness of our approach, we conduct evaluations on three LLMs LLaMA-7B (Touvron et al., 2023a), LLaMA-2-7B (Touvron et al., 2023b) and Mistral-7B-v0.1 (Jiang et al., 2023). We evaluate both NLP tasks and reasoning tasks. For the first setting, following (Gao et al., 2024), we assess the performance on three GLUE datasets and three commonsense reasoning datasets: (1) Microsoft's Research Paraphrase Corpus (MRPC) (Dolan and Brockett, 2005), (2) Recognizing Textual Entailment (RTE) dataset (Wang et al., 2019), (3) Corpus of Linguistic Acceptability (COLA) (Wang et al., 2019), (4) ScienceQA (Lu et al., 2022), (5) CommonsenseQA (Talmor et al., 2019), and (6) OpenbookQA (Mihaylov et al., 2018). For the second setting, we evaluate arithmetic reasoning on four zero-shot benchmarks: (1) AddSub (Hosseini et al., 2014), (2) MultiArith (Roy et al., 2015), (3) SVAMP (Patel et al., 2021), (4) GSM8K (Cobbe et al., 2021). The detailed description of each dataset is shown in Appendix B.

Baselines. We compare our method with MoLA, which involves the manual design of 4 different allocation strategies, with MoLA-▽(2468) as the state-of-the-art method. Specifically, take LLaMA-2 (Touvron et al., 2023b) which contains 32 layers, as an example. 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, which is denoted as **2468**. Thus, the overall structure forms a ▽ shape. Following similar notation, MoLA-□(5555) employs an allocation strategy of **5555**, uniformly assigning 5 experts to each layer. MoLA-□(8888) uniformly assigns 8 experts to each layer. To ensure a fair comparison, we introduce a target sum parameter T to control the total number of experts for `AlphaLoRA`, thereby equalizing the number of trainable parameters between `AlphaLoRA` and MoLA. The number of trainable

Models	Strategy	MRPC	COLA	RTE	S.QA	C.QA	O.QA	Average
LLaMA	**MoLA-□ (8888)**	82.55	84.37	84.47	90.82	76.82	76.60	82.61
	MoLA-□ (5555)	82.43	84.18	83.03	90.28	75.10	76.00	81.84
	MoLA-▽ (2468)	83.36	84.64	84.83	90.10	75.42	**78.60**	82.83
	AlphaLoRA	**85.19**	**85.42**	**85.19**	**90.37**	**76.49**	78.20	**83.48**
LLaMA-2	**MoLA-□ (8888)**	84.70	85.81	88.45	91.91	77.89	82.80	85.26
	MoLA-□ (5555)	84.17	86.19	84.83	92.08	77.55	80.00	84.14
	MoLA-▽ (2468)	83.48	**86.87**	86.28	92.36	**78.95**	79.60	84.59
	AlphaLoRA	**84.23**	86.67	**87.36**	**92.71**	78.05	**80.80**	**84.97**
Mistral	**MoLA-□ (8888)**	86.43	87.24	89.53	94.91	82.96	88.60	88.28
	MoLA-□ (5555)	85.73	87.34	88.44	94.46	81.90	88.00	87.65
	MoLA-▽ (2468)	86.95	87.44	88.80	**95.14**	83.37	88.20	88.32
	AlphaLoRA	**87.13**	**87.91**	**91.70**	95.00	**84.00**	**89.20**	**89.16**

Table 2: Accuracy comparison with different methods on direct fine-tuning (S.QA, C.QA, O.QA denote ScienceQA, CommonsenseQA, and OpenbookQA respectively). The total number of experts for MoLA-□ (8888) is 256, while the other variants are 160. AlphaLoRA outperforms other variants or baselines and even achieves competitive or superior performance with MoLA-□ (8888), with nearly 40% fewer parameters.

parameters is 105,635,840, which is 1.5% of the trainable parameters in the pre-trained base model.

4.1 Analysis of Expert Allocation

In Figure 3, we present the allocation of experts across all layers of the three LLMs: LLaMA-7B (Touvron et al., 2023a), LLaMA-2-7B (Touvron et al., 2023b), and Mistral-7B-v0.1 (Jiang et al., 2023). Our allocation indicates that the middle layers are generally better trained than the higher and lower layers, suggesting that they should be assigned fewer LoRA experts. In contrast, the higher layers are less well-trained and require more experts. Our approach reveals that the overall architecture of the three models forms a loosely "M" shape, a pattern not previously explored by (Gao et al., 2024). Additionally, we note that Mistral-7B-v0.1 requires more experts in the lower layers compared to the LLaMA models, highlighting the need for a model-specific allocation strategy.

4.2 Main Result

Direct fine-tuning. The first experimental setup adheres to the evaluation protocol detailed in (Gao et al., 2024). We perform direct instruction fine-tuning (Wei et al., 2021; Sanh et al., 2022) in various allocation strategies on six NLP datasets, assessing performance on their respective test sets.
① AlphaLoRA enhances efficiency in LoRA-MoE experts. In comparison to MoLA-□ (8888), which utilizes 256 experts, AlphaLoRA shows superior performance on LLaMA and Mistral models and achieves similar results on LLaMA-2, while using only 160 experts (62.5% of the parameters compared to MoLA-□ (8888)). For instance, AlphaLoRA surpasses MoLA-□ (8888) by 2.17% on the RTE dataset for the Mistral model. The results indicate that a uniform allocation strategy leads to redundancy across various models. AlphaLoRA ensures efficient allocation, enabling the experts to capture more knowledge.
② AlphaLoRA outperforms other baseline allocation methods. Given an equal total number of experts, AlphaLoRA, based on layer training quality, consistently outperforms MoLA-□ (5555) across three models, with performance improvements of 1.64%, 0.83%, and 1.51%, respectively. Furthermore, AlphaLoRA surpasses MoLA-▽ (2468) by 0.65%, 0.38%, and 0.84%, respectively. This highlights the superiority of our adaptive layer-wise allocation strategy.
③ Correlation between number of experts and layer training quality. As shown in Figure 3, the overall allocation of AlphaLoRA is more similar to MoLA-▽ (2468) compared to the difference with MoLA-□ (5555), resulting in a relatively smaller performance improvement. This underscores the correlation between the number of layer experts and training quality, suggesting that well-trained layers require fewer LoRA experts.

Models	Strategy	GSM8K	SVAMP	AddSub	MultiArith	Average
LLaMA	**MoLA-□ (5555)**	44.04	52.00	38.89	**88.16**	55.77
	MoLA-▽ (2468)	43.59	52.80	40.50	84.33	55.31
	AlphaLoRA	**45.03**	**53.60**	**42.27**	86.66	**56.89**
LLaMA-2	**MoLA-□ (5555)**	49.50	**57.10**	47.08	87.00	60.17
	MoLA-▽ (2468)	50.11	56.40	**48.86**	87.66	60.76
	AlphaLoRA	**50.41**	57.00	48.60	**91.33**	**61.84**
Mistral	**MoLA-□ (5555)**	69.37	73.60	56.45	96.00	73.86
	MoLA-▽ (2468)	67.20	76.50	59.24	97.00	74.99
	AlphaLoRA	**68.30**	**77.20**	**59.49**	**97.33**	**75.58**

Table 3: Accuracy comparison with different methods with same total experts number on zero-shot tasks evaluation. Each method is trained on the MetaMathQA dataset (Yu et al., 2024) and evaluated on four mathematical datasets.

Zero-shot tasks. In the second setting, we perform instruction-tuning on the MetaMathQA dataset (Yu et al., 2024) and evaluate on four zero-shot benchmarks: GSM8K (Cobbe et al., 2021), SVAMP (Patel et al., 2021), AddSub (Hosseini et al., 2014), and MultiArith (Roy et al., 2015). This setting evaluates the transfer learning capabilities of different allocation strategies. As shown in Table 3, AlphaLoRA outperforms both the state-of-the-art allocation strategy MoLA-▽ (2468) and the uniform allocation MoLA-□ (5555). For example, AlphaLoRA exceeds MoLA-▽ (2468) by an average of 1.58% across the four benchmarks. The results corroborate our earlier findings from direct fine-tuning experiments, demonstrating that a fine-grained allocation strategy like AlphaLoRA enables models to capture more knowledge and reduce redundancy among experts, thereby enhancing overall performance.

4.3 Layer Quality Metrics for Expert Allocation

In this study, we evaluate several shape metrics in HT-SR theory for measuring layer training quality. The definition of other metrics can be found in Appendix A. Experiments are conducted on LLaMA-2 (Touvron et al., 2023b) under the same parameter setting for each task. Figure 4 shows that PL_Alpha_Hill outperforms other shape metrics on both direct fine-tuning and zero-shot setting, aligning with the findings of (Zhou et al., 2024) that the PL_Alpha_Hill metric is better for assessing layer training quality. This further demonstrates that layer expert number correlates with layer training quality. The detailed results are shown in Table 6, 7 and 8, Appendix D.1.

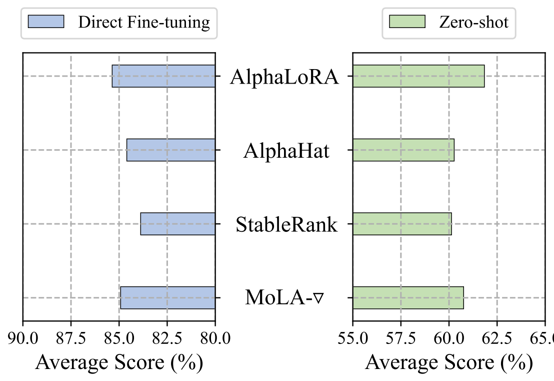

Figure 4: Comparison with different shape metric from HT-SR theory on both direct fine-tuning and zero-shot setting.

4.4 Consistent Superiority with Varying Number of Experts

In this study, we compare AlphaLoRA with MoLA-▽ across three configurations on Mistral-7B model, each with a different total number of experts T, specifically 80, 160, and 224 experts. We report the average score of 4 benchmarks, detailed results could be found in Table 5, Appendix D.2. In addition to MoLA-▽(2468), we introduce two variants that follow the MoLA-▽ pattern, featuring a gradually increasing number of experts from lower layers to higher layers. For instance, MoLA-▽(46810) allocates **4** experts to each layer for the first 8 layers, **6** experts to each layer for 9-16 layers, **8** experts to each layer for 17-24 layers, and **10** experts to each layer for the last 8 layers, which is denoted as **(46810)**. Figure 5 shows AlphaLoRA outperforms MoLA-▽ across three configurations. Notably, AlphaLoRA with 80 experts surpasses MoLA-▽(2468) with 160 experts, achieving comparable performance with 50% fewer parameters.

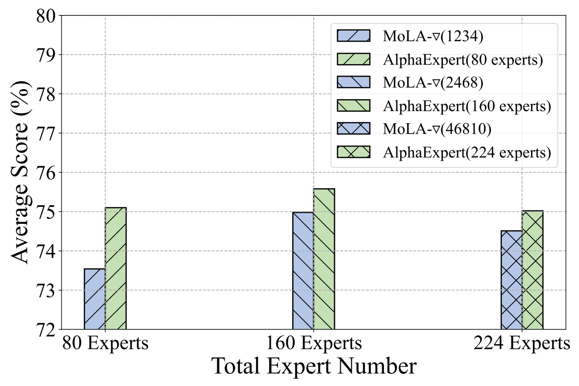

Figure 5: Comparison between AlphaLoRA and MoLA-▽ across three configurations with varying total number of experts T, specifically 80, 160, and 224 experts.

5 Related work

5.1 Parameter-Efficient Tuning

As language models continue to expand in size, parameter-efficient tuning of LLMs has attracted significant interest due to its cost-effective approach for fine-tuning. Researchers introduce several PEFT approaches, such as LoRA (Hu et al., 2022) and adapters (Houlsby et al., 2019), aimed at enhancing the efficiency of fine-tuning large models. Among these, PEFT techniques based on low-rank adapters, called LoRA, have gained significant popularity and widespread adoption. These methods introduce two trainable low-rank matrices within each fully connected layer, resulting in substantial savings in training resources while maintaining performance. Building on these ideas, our approach integrates the MoE technique with LoRA adapters, employing layer-wise expert allocation to further enhance performance.

5.2 LoRA-MoE Architecture

Recent research explores the integration of MoE (Shazeer et al., 2017b) and PEFT methods to boost performance in both single-task and multi-task scenarios (Li et al., 2024; Huang et al., 2023; Zhang et al., 2023; Yang et al., 2024; Dou et al., 2024; Feng et al., 2024; Liu et al., 2023; Luo et al., 2024). For instance, Liu et al. (2023) employs LoRA and MoE for multi-task scenarios, particularly in healthcare. However, their methods necessitate the data type as input, limiting the model's applicability to other tasks. Similarly, Dou et al. (2023) propose LoRAMoE, a novel adapter architecture that integrates MoE and LoRA within the feed-forward layer of each Transformer block, addressing the issue of knowledge forgetting in LLMs during traditional supervised fine-tuning. Nonetheless, these approaches uniformly initialize the number of LoRA experts for each layer, resulting in redundancy among LoRA experts. Gao et al. (2024) investigate redundancy in parameter-efficient MoE, initializing the number of experts with varying group-wise allocation and suggesting that higher layers require more LoRA experts. However, their allocation strategies are based on intuitive trial-and-error, lacking in-depth interpretability. Taking a step further, we introduce a fine-grained layer-wise expert allocation strategy by leveraging HT-SR theory to analyze layer training quality, improving the expert allocation in a theoretically principled manner.

5.3 Heavy-Tailed Self-Regularization Theory.

Martin and Mahoney (2019, 2020, 2021); Martin et al. (2021) explore the HT properties observed in the ESD of weight matrices in neural networks. These HT structures provide insights into the underlying behavior and quality of models. Recent studies further demonstrate the utility of HT-SR Theory in various aspects of deep learning (Yang et al., 2023; Zhou et al., 2024; Lu et al., 2024; Liu et al., 2024; Kothapalli et al., 2024). Yang et al. (2023) apply HT-SR principles to model selection, illustrating how the heavy-tailed characteristics can be leveraged to assess pre-trained NLP models without requiring training or testing data. Similarly, Zhou et al. (2024) extend this application to layer-wise adaptive training, showing that HT-SR can effectively guide the training process by assessing the quality of individual layers within a network. On the theory side, HT-SR theory has been substantially studied from various perspectives, including feature learning (Wang et al., 2024; Kothapalli et al., 2024) and overparameterization (Hodgkinson et al., 2023), which contribute to understanding the emergence of HT structures in the ESDs. Building on these insights, we utilize HT-SR theory to design an improved expert allocation method.

6 Conclusion

In this paper, we apply analytical methods from HT-SR theory to develop a fine-grained allocation strategy for determining the number of experts per layer in the LoRA-MoE architecture. Extensive empirical results show that AlphaLoRA offers a simple yet effective approach to layer-wise expert allocation. Analysis of three widely used transformer-based language models reveals that well-trained layers require fewer LoRA experts. Our theoretically grounded method provides a scalable solution for various models, further reducing redundancy within the LoRA-MoE architecture. In future work, we aim to integrate AlphaLoRA with other PEFT methods and explore its application across different model architectures and domains.

7 Limitations

AlphaLoRA demonstrates both effectiveness and scalability as a model-specific method for determining the number of LoRA experts. However, there are some potential limitations to consider. First, the performance of AlphaLoRA could be varied for different tasks, which could increase the uncertainty in the performance of this method. Additionally, the optimal total number of experts is determined by the experimental results. Overall, we will continue to work on this problem to address these limitations and develop more effective and robust allocation methods for different tasks.

Ethics Statement

We have not identified any ethical concerns directly related to this study.

Acknowledgment

This research was partially funded by a Google Research Award.

References

Zhidong Bai and Jack W Silverstein. 2010. *Spectral analysis of large dimensional random matrices*. Springer.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, and et al. 2020. Language models are few-shot learners. In *Neural Information Processing Systems*.

Tianlong Chen, Zhenyu Zhang, Ajay Jaiswal, Shiwei Liu, and Zhangyang Wang. 2023. Sparse moe as the new dropout: Scaling dense and self-slimmable transformers. In *International Conference on Learning Representations*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, and et al. 2022. Palm: Scaling language modeling with pathways.

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.

Romain Couillet and Zhenyu Liao. 2022. *Random matrix methods for machine learning*. Cambridge University Press.

Willian Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, et al. 2023. The art of balancing: Revolutionizing mixture of experts for maintaining world knowledge in language model alignment.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, et al. 2024. LoRAMoE: Alleviating world knowledge forgetting in large language models via MoE-style plugin. In *Association for Computational Linguistics*.

Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Yu Han, and Hao Wang. 2024. Mixture-of-LoRAs: An efficient multitask tuning method for large language models. In *Joint International Conference on Computational Linguistics, Language Resources and Evaluation*.

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.

Bruce M Hill. 1975. A simple general approach to inference about the tail of a distribution.

Liam Hodgkinson, Chris van der Heide, Robert Salomone, Fred Roosta, and Michael W Mahoney. 2023. The interpolating information criterion for overparameterized models. *arXiv preprint arXiv:2307.07785*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Empirical Methods in Natural Language Processing*.

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*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2023. Lorahub: Efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv:2307.13269*.

Yiren Jian, Tingkai Liu, Yunzhe Tao, Chunhui Zhang, Soroush Vosoughi, and Hongxia Yang. 2024. Expedited training of visual conditioned language generation via redundancy reduction. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers, Oral Presentation)*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, and et al. 2023. Mistral 7b.

Vignesh Kothapalli, Tianyu Pang, Shenyang Deng, Zongmin Liu, and Yaoqing Yang. 2024. Crafting heavy-tails in weight matrix spectrum without gradient noise. *arXiv preprint arXiv:2406.04657*.

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*.

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.

Zihang Liu, Yuanzhe Hu, Tianyu Pang, Yefan Zhou, Pu Ren, and Yaoqing Yang. 2024. Model balancing helps low-data training and fine-tuning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Haiquan Lu, Yefan Zhou, Shiwei Liu, Zhangyang Wang, Michael W. Mahoney, and Yaoqing Yang. 2024. Alphapruning: Using heavy-tailed self regularization theory for improved layer-wise pruning of large language models.

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 *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.

Charles H Martin and Michael W Mahoney. 2019. Traditional and heavy-tailed self regularization in neural network models. In *International Conference on Machine Learning*.

Charles H Martin and Michael W Mahoney. 2020. Heavy-tailed universality predicts trends in test accuracies for very large pre-trained deep neural networks. In *Proceedings of the 2020 SIAM International Conference on Data Mining*.

Charles H Martin and Michael W Mahoney. 2021. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning.

Charles H Martin, Tongsu Peng, and Michael W Mahoney. 2021. Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data.

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 *Empirical Methods in Natural Language Processing*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *Association for Computational Linguistics*.

Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about quantities in natural language.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2022. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017a. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017b. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *North American Chapter of the Association for Computational Linguistics*.

Terence Tao. 2023. *Topics in random matrix theory*. American Mathematical Society.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, and et al. 2023a. Llama: Open and efficient foundation language models.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, and et al. 2023b. Llama 2: Open foundation and fine-tuned chat models.

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

Zhichao Wang, Andrew Engel, Anand D Sarwate, Ioana Dumitriu, and Tony Chiang. 2024. Spectral evolution and invariance in linear-width neural networks. In *Advances in Neural Information Processing Systems*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

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.

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

Yaoqing Yang, Ryan Theisen, Liam Hodgkinson, Joseph E Gonzalez, Kannan Ramchandran, Charles H Martin, and Michael W Mahoney. 2023. Test accuracy vs. generalization gap: Model selection in nlp without accessing training or testing data. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, and James T et al. Kwok. 2024. Metamath: Bootstrap your own mathematical questions for large language models. In *International Conference on Learning Representations*.

Chunhui Zhang, Yiren Jian, Zhongyu Ouyang, and Soroush Vosoughi. 2024. Working memory identifies reasoning limits in language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Jinghan Zhang, Junteng Liu, Junxian He, et al. 2023. Composing parameter-efficient modules with arithmetic operation. In *Advances in Neural Information Processing Systems*.

Yefan Zhou, Tianyu Pang, Keqin Liu, Michael W Mahoney, Yaoqing Yang, et al. 2024. Temperature balancing, layer-wise weight analysis, and neural network training.

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.

A Definition of other shape metrics from HT-SR theory

- (Stable_Rank) The Stable_Rank metric provides a norm-adjusted assessment of the scale of the empirical spectral density (ESD). Prior research (Martin et al., 2021) has shown that Stable_Rank correlates with the PL_Alpha metric. For a given weight matrix $\mathbf{W}$, it is calculated as follows:

$$\texttt{Stable\_Rank} = \frac{\|\mathbf{W}\|_F^2}{\|\mathbf{W}\|_2^2}, \tag{10}$$

  where $\|\mathbf{W}\|_F$ denotes the Frobenius norm and $\|\mathbf{W}\|_2$ denotes the spectral norm.

- (Alpha_Hat) The Alpha_Hat metric, introduced in (Martin et al., 2021), has been demonstrated to be effective at predicting model generalization. It represents a modified form of the Power Law (PL) exponent α (PL_Alpha), scaled by the logarithm of the largest eigenvalue $\lambda^{\max}$ of the weight matrix's spectral norm (log_spectral_norm):

$$\texttt{Alpha\_Hat} = \alpha \log \lambda^{\max}. \tag{11}$$

B Dataset

AlphaLoRA is studied on ten standard datasets from three categories:

B.1 Language Understanding

Microsoft's Research Paraphrase Corpus (MRPC). This dataset has 5,801 sentence pairs from news articles, labeled to indicate paraphrases. It includes 4,076 pairs for training and 1,725 for testing, with the task of classifying paraphrase pairs.

Recognizing Textual Entailment (RTE). Derived from annual textual entailment challenges (RTE1, RTE2, RTE3, RTE5), this dataset features sentences from news and Wikipedia, classified as entailment or not. It comprises 2,490 training and 277 validation samples.

Corpus of Linguistic Acceptability (COLA). This dataset contains English sentences annotated for grammatical acceptability, sourced from linguistic theory texts. It includes 8,551 training and 1,043 validation samples.

B.2 Commonsense Reasoning

ScienceQA. This dataset includes 21,208 multiple-choice questions from elementary and high school science curricula. For text-only samples, there are 6,508 training and 2,224 test samples, covering natural science, language science, and social science, requiring commonsense knowledge for answers.

CommonsenseQA. A dataset for commonsense reasoning with 9,740 training samples and 1,221 validation samples, created by Amazon Mechanical Turk workers. It demands various types of commonsense knowledge to determine the correct answers.

OpenbookQA. Comprising 5,957 elementary-level science questions, this dataset tests understanding of core science facts and their application to new scenarios. It includes 4,957 training, 500 validation, and 500 test samples.

B.3 Arithmetic Reasoning

Table 4 presents the statistics of four Arithmetic Reasoning benchmarks: (1)AddSub (Hosseini et al., 2014), (2) MultiArith (Roy et al., 2015), (3) SVAMP (Patel et al., 2021), (4) GSM8K (Cobbe et al., 2021).

Dataset	# of Samples	Avg. Words
AddSub	395	31.5
MultiArith	600	31.8
SVAMP	1000	31.8
GSM8K	1319	46.9

Table 4: Statistics of Arithmetic Reasoning datasets.

C Implementation

The direct fine-tuning setting aligns with Gao et al. (2024), we do a grid search on the number of training epochs, including 10, 15, and 20 epochs for downstream task fine-tuning. The cutoff length is set to 256 and the batch size is 128. For the second instruction-tuning$\rightarrow$zero-shot tasks evaluation, we conduct instruction-tuning on the MetaMathQA dataset (Yu et al., 2024) for 1 epoch with cutoff length set to 512. We conduct a small hyperparameter sweep within the range of $\beta \in [2.0, 2.5, 3.0]$, where β regulates the standard deviation of experts allocation, as depicted in Figure 6. For all experiments, we use AdamW (Loshchilov and Hutter, 2017) as the optimizer with a learning rate of 3e-4.

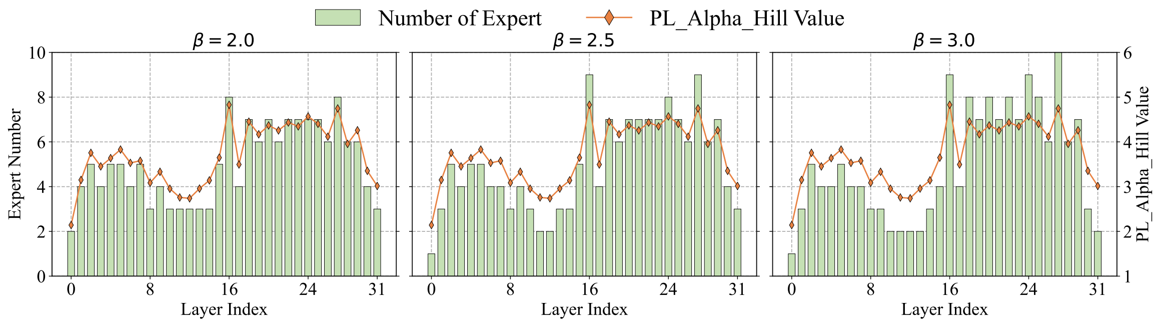

Figure 6: The expert allocations for Mistral-7b under different parameter β.

Expert Amount	Strategy	GSM8K	SVAMP	AddSub	MultiArith
80	**MoLA-▽(1234)**	68.23	73.90	56.70	95.33
	AlphaLoRA	**69.44**	**75.60**	**57.72**	**97.66**
160	**MoLA-▽ (2468)**	67.20	76.50	59.24	97.00
	AlphaLoRA	**68.30**	**77.20**	**59.49**	**97.33**
224	**MoLA-▽(46810)**	68.61	73.60	**58.98**	96.83
	AlphaLoRA	**69.06**	**75.30**	58.22	**97.50**

Table 5: Comparison between `AlphaLoRA` and MoLA-▽ across three configurations with varying total number of experts T, specifically 80, 160, and 224 experts.

Strategy	MRPC	COLA	RTE
MoLA-▽	83.48	**86.87**	86.28
Stable_Rank	**84.34**	84.56	86.64
Alpha_Hat	84.11	86.86	84.83
AlphaLoRA	84.23	86.67	**87.36**

Table 6: Comparison of shape metrics on GLUE tasks.

Strategy	S.QA	C.QA	O.QA
MoLA-▽	92.36	**78.95**	79.60
Stable_Rank	92.13	77.81	77.80
Alpha_Hat	91.86	78.13	79.80
AlphaLoRA	**92.71**	78.37	**80.80**

Table 7: Comparison of shape metrics on QA tasks (S.QA, C.QA, O.QA denote ScienceQA, CommonsenseQA, and OpenbookQA respectively).

The rank of each LoRA expert is 8 and we adopt Top-2 for the router. LoRA alpha is set to 16 and LoRA dropout is 0.05, following the default LoRA settings (Hu et al., 2022). We apply LoRA experts to four weight matrices in the self-attention module ($\mathbf{W_q}$, $\mathbf{W_k}$, $\mathbf{W_v}$, $\mathbf{W_o}$) and three weight matrices in the MLP module ($\mathbf{W_{gate}}$, $\mathbf{W_{down}}$, $\mathbf{W_{up}}$). All experiments are conducted with three RTX A6000-48G GPUs.

Shape Metrics	GSM8K	SVAMP	AddSub	MultiArith
MoLA-▽	50.11	56.40	**48.86**	87.66
Stable_Rank	48.22	55.20	48.61	88.50
Alpha_Hat	49.81	56.70	46.08	88.50
AlphaLoRA	**50.41**	**57.00**	48.60	**91.33**

Table 8: Comparison of shape metrics on arithmetic tasks.

D Complementary Results

In this section, we provide detailed results in Section 4.3 and Section 4.4.

D.1 Detailed Results of Different Shape Metric

Table 6 and 7 present the results of several shape metrics in HT-SR theory when directly fine-tuned on GLUE and QA tasks. Table 8 shows the results of zero-shot setting on math tasks. We report the average score for both settings in Figure 4.

D.2 Detailed Results of Varying Number of Experts

In Table 5, we show the results of `AlphaLoRA` and MoLA-▽ across three configurations with varying total number of experts T, specifically 80, 160, and 224 experts.