✂️LoRA 本质是朴素的矩阵分解思想，和 DeepFM、协同推荐、模型剪枝 / 压缩等有相通之处。
在 PEFT 库中实现，也是简单直觉，对于全连接层 (Linear)，就是伴生两个小矩阵。
具体来说，配置解析在 LoraModel（图 4），伴生矩阵创建在基类 LoraLayer，调用在自己的 Linear 实现。
-
✅有三个小细节：
-
0️⃣ PEFT 中两个伴随矩阵的初始化，一个是高斯，一个是零阵，实现在 reset_lora_parameters；scale 是论文中的 alpha /r。
1️⃣ 运算其实就一行 `result += lora_B (lora_A (dropout (x))) * scaling`，这里有个优化点，scaling 在运算中的位置会影响计算量，你认为放哪更优呢？
2️⃣ 在上图中能看出，不仅支持 Linear，还支持 Embedding、Conv2d，我们下回贴实现。
-
✨DoRA 实现细节
❗权重分解：对于模型中的每个权重张量，DoRA 首先将其分解为幅度和方向两个部分。通过计算权重张量的 Frobenius 范数来得到幅度，而方向则由单位化的原权重张量给出。
❗方向更新：在微调过程中，DoRA 专注于对权重的方向部分进行更新。这是通过引入额外的低秩矩阵完成的，这些矩阵与原始的方向矩阵相乘，从而实现对方向的精细调整。
❗幅度调整：虽然 DoRA 的重点是方向更新，但它也允许对幅度进行细微的调整，以进一步优化模型性能。这种调整通常较小，目的是保持模型的一般能力，同时对特定任务进行特定的优化。
❗优化与正则化：为了确保微调过程的稳定性，并避免过拟合，DoRA 采用了特定的优化策略和正则化技术。这些技术旨在平衡模型在保持预训练知识和适应新任务之间的能力。
-
✨实验结果
DoRA 在提升模型性能方面具有显著优势。在常识推理、视觉指令调整和图像 / 视频 - 文本理解等任务上，DoRA 相比于 LoRA 和其他 PEFT 方法显示出一致的性能提升。通过与 FT 比较，DoRA 能够在保持较低的额外推理成本的同时，接近甚至达到全参数微调的性能。

Higher Layers Need More LoRA Experts

Chongyang Gao [1] Kezhen Chen [2] Jinmeng Rao [2] Baochen Sun [2] Ruibo Liu [3] Daiyi Peng [3] Yawen Zhang [2] Xiaoyuan Guo [2] Jie Yang [2] VS Subrahmanian [1]

Abstract

Parameter-efficient tuning (PEFT) techniques like low-rank adaptation (LoRA) offer training efficiency on Large Language Models, but their impact on model performance remains limited. Recent efforts integrate LoRA and Mixture-of-Experts (MoE) to improve the performance of PEFT methods. Despite promising results, research on improving the efficiency of LoRA with MoE is still in its early stages. Recent studies have shown that experts in the MoE architecture have different strengths and also exhibit some redundancy. Does this statement also apply to parameter-efficient MoE? In this paper, we introduce a novel parameter-efficient MoE method, ***M**oE-**L**o**RA** with **L**ayer-wise Expert **A**llocation (MoLA)* for Transformer-based models, where each model layer has the flexibility to employ a varying number of LoRA experts. We investigate several architectures with varying layer-wise expert configurations. Experiments on six well-known NLP and commonsense QA benchmarks demonstrate that MoLA achieves equal or superior performance compared to all baselines. We find that allocating more LoRA experts to higher layers further enhances the effectiveness of models with a certain number of experts in total. With much fewer parameters, this allocation strategy outperforms the setting with the same number of experts in every layer. This work can be widely used as a plug-and-play parameter-efficient tuning approach for various applications. The code is available at `https://github.com/GCYZSL/MoLA`.

[1]Department of Computer Science, Northwestern University, Evanston, IL [2]Mineral Research, USA [3]Google DeepMind, USA. Correspondence to: Chongyang Gao <chongyanggao2026@u.northwestern.edu>, Kezhen Chen <kzhen0204@gmail.com>, VS Subrahmanian <vss@northwestern.edu>.

1. Introduction

Large Language Models (LLMs) have shown impressive proficiency and transfer learning capabilities across a variety of tasks and domains (Chowdhery et al., 2022; Zhang et al., 2023b; Anil et al., 2023; Jiang et al., 2024; Singhal et al., 2022; Zhu et al., 2023; Lv et al., 2023). However, modern LLMs fine-tuning demands huge computational resources due to the vast number of parameters. To mitigate this issue, the research community is increasingly focusing on parameter-efficient fine-tuning (PEFT) methods to dramatically reduce training costs, such as p-tuning (Liu et al., 2022b) or low-rank adaption (LoRA) (Hu et al., 2022). Despite its training efficiency, PEFT methods' performance in fine-tuning LLMs is still limited.

Recent studies show that combining PEFT with the Mixture-of-Experts (MoE) becomes a promising recipe for leveraging MoE in a parameter-efficient fashion (Zadouri et al., 2023; Liu et al., 2023; Dou et al., 2023), providing impressive performance. Most of these methods apply MoE on LoRA, called LoRA-MoE. For Transformer models (Vaswani et al., 2017), LoRA learns a pair of low-rank matrices as an adapter for a given dense linear layer, effectively modifying the layer's behavior without substantial change to the original model parameters. Instead of learning one pair of low-rank matrices, LoRA-MoE learns multiple pairs of low-rank matrices, called *LoRA experts*, and a router to compute the weights of each expert for inputs. During the LLM fine-tuning phase, pre-trained weights of dense layers remain fixed, while LoRA experts and the router are trained to adapt the pre-trained weights. While the initial results are promising, the research into achieving more efficient and effective integration is still in its infancy.

Moreover, recent studies in the MoE analysis indicate that the use of many experts may be redundant due to representational collapse or learned routing policy overfitting (Chen et al., 2023; Zoph et al., 2022). More experts in a layer may cause the representation to overfit the training data, as the data is processed in a more fine-grained manner. This insight leads us to think about how many experts could be more suitable for different layers in the Transformer model, motivating us to explore two questions.

(i) Are there any redundant experts in parameter-efficient

MoE? (ii) What strategy should be used to allocate the number of LoRA experts in each layer?

To address these questions, we introduce a *new* parameter-efficient MoE approach, ***M**oE-**Lo**RA with **L**ayer-wise Expert **A**llocation (MoLA)*, combining LoRA and MoE with layer-wise expert allocation. Users can flexibly assign a different number of LoRA experts to each Transformer layer. We study several typical architectures with different layer-wise expert configurations. Using a fixed number of experts in total, we allocate them differently, with either lower layers or higher layers having more experts. We conduct experiments on six benchmarks including NLP and commonsense question-answering tasks to demonstrate the effectiveness of our MoLA approach under different configurations.

Key Findings: Our extensive experiments reveal that experts in lower layers are more similar to each other and thus exhibit more redundancy. With a fixed number of experts, more LoRA experts should be allocated to the higher layers of the Transformer model to enhance its effectiveness. Our key contributions are:

- We present a new parameter-efficient MoE method, MoLA, with flexible layer-wise expert allocation, on the Transformer model. MoLA integrates LoRA and MoE and introduces flexibility to assign different numbers of experts to different Transformer layers, reducing expert redundancy and diversifying information granularity. MoLA is a plug-and-play approach and can be applied to diverse tasks.
- We study several MoLA variants on an LLM, each with different layer-wise expert configurations. Experiments on six benchmarks show that all MoLA configurations significantly outperform other PEFT baselines, showing the efficacy of our approach.
- We further compare each layer-wise configuration of expert allocation. *Overall, the configuration, that has more LoRA experts in the higher layers and fewer in the lower layer, outperforms all other configurations.* Such specialized expert allocation configuration enables models to achieve enhanced performance vis-a-vis other configurations, even with much fewer parameters, demonstrating improved scalability.
- Our comprehensive analysis shows that experts in lower layers are more similar than those in higher layers and thus have higher redundancy, providing insights into our observations.

2. Preliminaries

We first briefly review MoE and LoRA before describing our MoLA framework.

Mixture of Experts The MoE architecture (Shazeer et al., 2017) applies sparse sub-modules, called experts, to various inputs via a router module. The router module intelligently employs different experts for different types of inputs, thus scaling up model parameters with a constant computational cost. MoE has shown promising effectiveness on the Transformer model (Shazeer et al., 2017). The MoE layer consists of N identical and independent feed-forward neural networks $\{E\}_{i=1}^{N}$ as experts. The router is a gating function with a trainable weight matrix W_r. Given an input x, the router maps x to an N-dimensional vector, which corresponds to the number of experts. The router uses a softmax function to compute a probability distribution of the weights of outputs from the expert networks. Following standard MoE architectures, only the top K experts, determined by the router, are chosen for the computation. Additionally, an auxiliary loss, called load balancing loss, is used on each MoE layer to promote a balanced top-k selection by pushing the router to have equitable workload distribution among experts. Equation 1 mathematically represents the MoE layer where y is the output embedding from the MoE layer. With fine-tuning, different experts focus on processing different types of information or tasks and thus provide finer granularity.

$$y = \sum_{i=1}^{K} \frac{\text{TopK}(\text{Softmax}(W_r x), K)_i}{\sum_{i=1}^{K} \text{TopK}(\text{Softmax}(W_r x), K)_i} E_i(x) \tag{1}$$

LoRA LoRA is a popular parameter-efficient tuning approach that is widely used in LLM fine-tuning(Hu et al., 2022; Zhang et al., 2023a; Dettmers et al., 2023). LoRA leverages low-rank matrix decomposition of pre-trained weight matrices to significantly reduce the number of training parameters. Given a pre-trained linear layer with a weight matrix $W_0 \in \mathbb{R}^{d_q \times d_p}$, LoRA creates two low-rank trainable matrices A and B, where $A \in \mathbb{R}^{d_q \times r}$, $B \in \mathbb{R}^{r \times d_p}$, and $r \ll min(d_q, d_p)$. Thus, the dimension of ABx equals the dimension of $W_0 x$. Equation 2 mathematically describes this process and the output of LoRA is h. During training, W_0 is frozen and does not receive gradient updates, while A and B are updated.

$$h = W_0 x + \triangle W x = W_0 x + ABx \tag{2}$$

The matrix A is initialized with a random Gaussian distribution and matrix B is initialized to zero. The initialization results in the same outputs as the original pre-trained model. When fine-tuning LLMs, the LoRA approach can be applied to all the linear layers in the Transformer model or its variants. Compared with tuning the original weight matrix, LoRA dramatically reduces the number of training parameters while keeping reasonable performance.

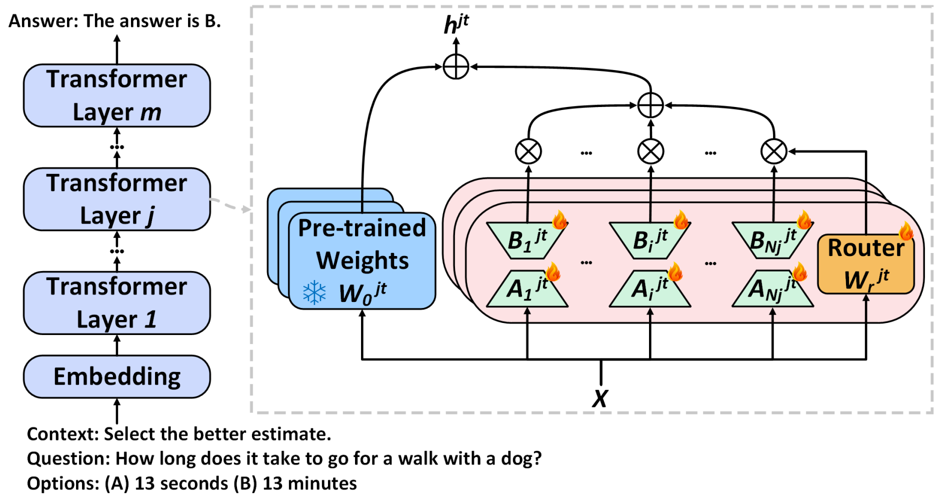
Answer: The answer is B.
Transformer Layer m
Transformer Layer j
Transformer Layer 1
Embedding
Context: Select the better estimate.
Question: How long does it take to go for a walk with a dog?
Options: (A) 13 seconds (B) 13 minutes
h^jt
Pre-trained Weights W_0^jt
B_1^jt
B_i^jt
B_Nj^jt
A_1^jt
A_i^jt
A_Nj^jt
Router W_r^jt
X

Figure 1. The overview of MoLA architecture. MoLA applies LoRA-MoE on a pre-trained Transformer model with layer-wise expert allocation. Each layer employs a different number of experts. During training, the pre-trained weights are freeze and only LoRA experts are tuned as the adapters on the weights.

3. MoE-LoRA with Layer-wise Allocation

Combining MoE and LoRA has shown promising results (Zadouri et al., 2023; Liu et al., 2023; Dou et al., 2023). However, most such efforts only replace experts with LoRA adapters under the MoE framework, and each layer has a fixed number of experts. Thus, some shortcomings of MoE may persist in these methods. For instance, experts in MoE may be redundant due to representational collapse or learned routing policy overfitting (Chen et al., 2023; Zoph et al., 2022). Inspired by this insight, we argue that the number of LoRA experts need *not* be the same across all Transformer layers.

We thus introduce a novel parameter-efficient tuning approach, called MoE-LoRA with Layer-wise Allocation (MoLA), which combines LoRA and MoE techniques with smart layer-wise expert allocation. As most LLMs use Transformer-based architectures, we study how MoLA should be applied to the Transformer model. Instead of allocating the same number of experts to all layers of the Transformer, MoLA uses different numbers of experts on different layers. In this section, we first describe the details of our architecture and then propose several layer-wise expert allocations based on different assumptions.

3.1. The MoLA Architecture

MoLA integrates LoRA adapters into the MoE framework so each layer may have a different number of experts. When training a pre-trained LLM with LoRA, instead of decomposing each weight matrix of a dense linear layer into a pair of low-rank matrices, we create *multiple* pairs of low-rank matrices — each pair is called a LoRA expert. A router module is learned to route each input token to different LoRA experts. Given a Transformer model with m layers, we allocate N_j experts for layer j and have $\sum_{j=1}^{m} N_j$ experts in total. Specifically, given a pre-trained weight matrix $W_0^{jt} \in \mathbb{R}^{d_q \times d_p}$ from the module t in layer j, we create N_j pairs of low-rank matrices $\{A^{jt}\}_{i=0}^{N_j}, \{B^{jt}\}_{i=0}^{N_j}$. As in the case of LoRA, each matrix A_i^{jt} is initialized from a random Gaussian distribution. We set B_i^{jt} to zero, where $A_i^{jt} \in \mathbb{R}^{d_q \times r}$, $B_i^{jt} \in \mathbb{R}^{r \times d_p}$, and $r \ll min(d_q, d_p)$. Then, a router S_i^{jt} with a trainable weight matrix $W_r^{jt} \in \mathbb{R}^{d_q \times N_j}$ is used to specify different LoRA experts for the input x. As in the original MoE, MoLA selects the top K experts for computation and applies the load balancing loss on each layer. Figure 1 shows an overview of the architecture. The mathematical representation is:

$$S_i^{jt}(x) = \frac{\text{TopK}(\text{Softmax}(W_r^{jt}x), K)_i}{\sum_{i=1}^{K} \text{TopK}(\text{Softmax}(W_r^{jt}x), K)_i} \tag{3}$$

$$h^{jt} = W_0^{jt}x + \sum_{i=1}^{K} S_i^{jt}(x) A_i^{jt} B_i^{jt} x \tag{4}$$

Eq. 3 represents the router with the input x and Eq. 4 mathematically shows the LoRA expert in MoLA, where h^{jt} is the output embedding. This MoLA architecture provides the flexibility to modify the number of experts for each Transformer layer. The next section addresses the question of how experts should be allocated in each layer.

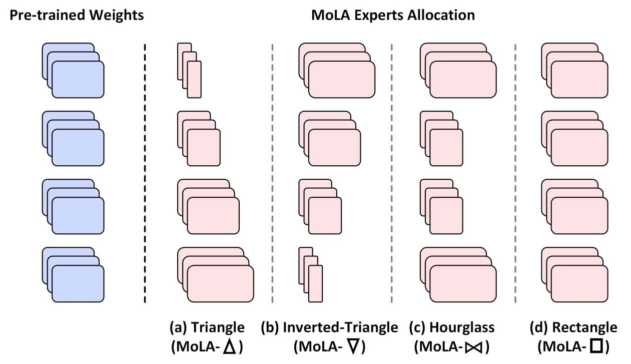

Figure 2. Four types of layer-wise expert allocations of MoLA.

3.2. Configurations of Layer-wise Expert Allocation

MoE is similar to an ensemble method with multiple experts learning fine-grained information. Layers with more experts have stronger fitting capabilities. One intuition is that we should allocate more experts to layers that are required to process diverse edge cases and fine-grained information. To study how LoRA experts should be allocated in each Transformer layer, we propose four types of layer-wise expert configurations based on different assumptions. Figure 2 visualizes the overview of these four configurations. Section 4 describes detailed experiments to compare these configurations.

MoLA Triangle (MoLA-△) Many studies have analyzed layer-wise representations of Transformer models. Generally, lower layers learn more token-level features, such as word meaning, syntax, or grammar, while higher layers capture more abstract, high-level information. As token-level information is subtle and diverse, one assumption is that token-level information may require more experts to distinguish fine-grained meaning while high-level information may require fewer experts for generalization. Our MoLA Triangle (MoLA-△) architecture is based on this assumption and allocates experts in a "triangle" shape: lower layers have more experts than higher layers.

MoLA Inverted-Triangle (MoLA-▽) Unlike MoLA-△, another assumption is that more experts for processing token-level information may introduce redundancy during the information processing. As higher layers learn more abstract and high-level information, and these features are used for downstream tasks, they may require more experts. More experts may enhance the architecture to process complicated problems by leveraging experts to learn fine-grained and task-specific patterns. Based on this intuition, we design the MoLA Inverted-Triangle (MoLA-▽) configuration where lower layers are allocated fewer experts while higher layers have more experts.

MoLA Hourglass (MoLA-⋈) A third model assumes that both lower and higher layers require more experts as they focus on processing basic features and abstract features. The middle layers play a role in aggregating the basic features and mapping them to a high-dimensional space for abstract reasoning, requiring fewer fine-grained features. Our MoLA Hourglass (MoLA-⋈) architecture uses this assumption to allocate experts in an "hourglass" shape, where lower and higher layers have more experts than the middle layers.

MoLA Rectangle (MoLA-□) The last configuration is the original design of MoE, where each Transformer layer has the same number of experts. Most of the recent studies adopt this expert allocation design. We call this MoLA Rectangle (MoLA-□) and use it as a baseline.

4. Experiments

4.1. Experiment Settings

We designed two experimental settings to examine the performance of our proposed MoLA approach, including instruction-tuning→fine-tuning and direct fine-tuning. To make the comparisons straightforward and clear, we designed 4 allocation configurations of MoLA for the large language model as illustrated in Section 3.2. We take LLaMA-2 (Touvron et al., 2023) which contains 32 layers as our base model. For MoLA-△, we allocate **8** experts to each layer for the first 8 layers, **6** experts to each layer for the next 8 layers, **4** experts to each layer for 17-24 layers, and **2** experts to each layer for the last 8 layers, which is denoted as **8642**. Following the same notation, we allocate MoLA Inverted Triangle as **2468**. The allocations for Hourglass and MoLA Rectangle are **8228** and **5555**, separately. Notably, to make the comparison fair, we make the total number of experts the same for all the variants, i.e., the same number of trainable parameters. The trainable parameter number is 105,635,840, which is a 1.5% trainable parameter number of the pre-trained base model.

In the first setting, we perform instruction tuning (Wei et al., 2021; Sanh et al., 2022; Mishra et al., 2022) with PEFT methods on an instruction-tuning dataset with cross-entropy loss. We also adopt auxiliary loss for balancing the top-k selection of routing following Switch Transformers (Fedus et al., 2022). We then fine-tune the pre-trained PEFT models on downstream tasks. In the second setting, we directly fine-tune the PEFT models on downstream tasks without instruction tuning.

4.2. Task and Data

MoLA is intended to fine-tune LLMs on downstream tasks and/or fine-tune instructions. To show its effectiveness, we study both natural language processing (NLP) tasks and commonsense reasoning (CR) tasks. For NLP tasks, we evaluate three popular datasets, including Microsoft's Research Paraphrase Corpus (Dolan & Brockett, 2005), Recognizing Textual Entailment (RTE) dataset (Wang et al., 2019), and Corpus of Linguistic Acceptability (COLA) (Wang et al., 2019). For commonsense reasoning tasks, we evaluate three recent question-answering benchmarks, including ScienceQA (Lu et al., 2022), CommonsenseQA (Talmor et al., 2019), and OpenbookQA (Mihaylov et al., 2018). We follow the task-specific fine-tuning framework to evaluate their effectiveness.

Microsoft's Research Paraphrase Corpus (MRPC) consists of 5,801 sentence pairs collected from newswire articles. Each pair is labeled by whether it is a paraphrase or not and the task is to classify whether sentence pairs are paraphrases. The dataset is divided into a training set with 4,076 sentence pairs and a testing set with 1,725 pairs.

The Recognizing Textual Entailment (RTE) dataset comes from a series of annual textual entailment challenges including RTE1, RET2, RTE3, and RTE5. Sentence examples are constructed based on news and Wikipedia text. This dataset is a two-class classification, entailment or not entailment, containing 2,490 training and 277 validation samples.

The Corpus of Linguistic Acceptability (COLA) consists of English acceptability judgments drawn from books and journal articles on linguistic theory. Each example is a sequence of words annotated with whether it is a grammatical English sentence. This corpus has 8,551 training samples and 1,043 validation samples for checking grammar.

ScienceQA is a commonsense question-answering dataset collected from elementary and high school science curricula containing 21,208 multimodal multiple-choice sentence questions. Because this paper focuses on textual inputs, we gathered all text-only samples and created a training and test set of 6508 and 2224 samples, respectively. ScienceQA has rich diversity from three subjects: natural science and language science, social science. To answer these questions, models need to align with correct commonsense knowledge.

CommonsenseQA is a commonsense reasoning question-answering dataset that requires different types of commonsense knowledge to predict the correct answers. The dataset was generated by Amazon Mechanical Turk workers and contains 9,740 training samples and 1,221 validation samples.

OpenbookQA is a commonsense question-answering dataset for assessing human understanding of a subject. It consists of 5,957 multiple-choice elementary-level science questions with 4,957 training, 500 validation, and 500 test samples. To answer a question, models must probe the understanding of a small "book" of 1,326 core science facts and the application of these facts to novel situations.

Besides all the evaluation benchmarks, we also use an instruction-tuning corpus for training. We randomly sampled 50,000 samples from the OpenOrca (Lian et al., 2023) corpus. OpenOrca is an instruction-tuning dataset consisting of augmented FLAN data aligned with the distributions demonstrated in ORCA (Mukherjee et al., 2023) and it has 2.91M data samples across diverse tasks or instructions.

4.3. Recent Competitive Baselines

We compare MoLA with three parameter-efficient tuning approaches, prompt tuning (Lester et al., 2021), LLaMA-Adapter (Zhang et al., 2023b), and LoRA(Hu et al., 2022). We also evaluate full-parameter fine-tuning. Prompt tuning presents soft prompting concatenated to the embedding layer of the Transformer model. Soft prompts are a set of virtual tokens pre-appended to the textual prompt and passed to the LLM. During fine-tuning, the LLM is frozen and only the virtual tokens are optimized, providing a lightweight tuning approach. LLaMA-Adapter is an adaption method for LLaMA instruction tuning and has a set of learnable adaption prompts that are pre-appended to the word tokens at higher transformer layers. A zero-initialized attention mechanism with zero gating is used to inject new instructional cues into LLaMA. LoRA was briefly described in Section 3. Specifically, the rank of LoRA is 64. In our evaluation, LLMs are fine-tuned on the downstream training dataset via different parameter-efficient tuning approaches. Based on the availability of test set labels, we evaluated COLA, RTE, and CommonsenseQA on their validation set and others on the test set.

4.4. Implementation

We use LLAMA2-7B (Touvron et al., 2023) as our base language model across all the experiments. In the first setting, we trained the PEFT model on a sampled instruction-tuning dataset for 3 epochs. In both settings, we do a grid search on the number of training epochs, including 10, 15, and 20 epochs for downstream task fine-tuning. We use AdamW (Loshchilov & Hutter, 2017) as the optimizer with a learning rate of 3e-4. The cutoff length is set to 256 following Sanh et al. (2022) and the batch size is 128. 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. We applied LoRA to four weight matrices in the self-attention module (W_q , W_k, W_v , W_o) and three weight matrices in the MLP module (W_{gate}, W_{down}, W_{up}). All experiments were conducted on the servers with eight A100-40G GPUs.

4.5. Results

Comparison with Baselines Table 1 shows the results for the direct fine-tuning setting where each number is

Table 1. Comparison with different methods on directly downstream fine-tuning. MoLA-▽ outperforms other variants or baselines and even achieves competitive or superior performance with MoLA-□ (8888), with nearly 40% fewer parameters.

Models (# of Experts)	MRPC	COLA	RTE	ScienceQA	CommonsenseQA	OpenbookQA
Full-Parameter	87.13%	86.29%	87.73%	93.12%	77.48%	80.4%
Prompt Tuning	49.91%	59.25%	54.17%	36.78%	37.76%	46.2%
LLaMA-Adapter	71.94%	47.56%	72.93%	73.33%	73.55%	71.8%
LoRA	83.13%	**86.29%**	85.92%	91.01%	75.51%	77.0%
MoLA-□ (8888)	**84.70%**	85.81%	**88.45%**	**91.91%**	**77.89%**	**82.8%**
MoLA-□ (5555)	84.23%	86.28%	85.20%	92.04%	78.13%	**80.0%**
MoLA-△ (8642)	**84.64%**	85.43%	84.84%	91.90%	77.23%	77.6%
MoLA-⋈ (8228)	83.48%	86.00%	**86.28%**	91.41%	76.25%	78.8%
MoLA-▽ (2468)	83.48%	**86.87%**	**86.28%**	**92.36%**	**78.95%**	79.6%

Table 2. Comparison with different methods on instruction-tuning & downstream fine-tuning. MoLA-▽ outperforms other variants and shows promising transfer learning capability.

Models (# of Experts)	MRPC	COLA	RTE	ScienceQA	CommonsenseQA	OpenbookQA
LoRA	**84.41%**	84.95%	84.48%	91.01%	74.61%	76.6%
MoLA-□ (8888)	84.23%	**85.72%**	**87.36%**	**92.13%**	**77.15%**	**78.4%**
MoLA-□ (5555)	84.93%	84.56%	88.81%	91.73%	75.92%	77.6%
MoLA-△ (8642)	84.46%	85.23%	**89.17%**	91.41%	76.33%	**78.8%**
MoLA-⋈ (8228)	84.35%	84.85%	87.72%	91.41%	75.02%	77.4%
MoLA-▽ (2468)	**85.45%**	**86.19%**	**89.17%**	**92.36%**	**77.15%**	78.4%

the accuracy (%) for each dataset. From the table, LoRA-based approaches (LoRA and MoLA) significantly outperform prompt-tuning-based baselines (Prompt Tuning and LLaMA-Adapter). For LoRA-based methods, the original LoRA with rank 64 is used as our baseline. We first evaluate the MoLA-□ with eight experts at each layer, annotated as MoLA-□(8888), where the number of parameters is the same as the LoRA baseline. Then, we reduce the sum of configuration number from 32 (8×4) to 20 in total, with only 62.5% of the parameters, and evaluate the four different configurations as described in Section 4.1. MoLA variants outperform the LoRA baseline on all the benchmarks. Specifically, MoLA-▽ beats LoRA on all six datasets — the performance improvements of MoLA-▽ are larger on the commonsense QA tasks compared to the NLP tasks. It even outperforms the MoLA-□(8888) on three benchmarks with nearly 40% fewer parameters. The results demonstrate the effectiveness and scalability of MoLA.

Table. 2 presents the results, in accuracy(%), for the instruction-tuning→fine-tuning setting. The language model is first tuned via each PEFT approach on our instruction-tuning set. The model is then fine-tuned on all downstream tasks. This setting evaluates the transfer learning capability of each PEFT approach. We only compare the LoRA-based methods due to their superior transfer learning capabilities (vs. prompt-tuning-based methods). Our results show that MoLA variants significantly outperform LoRA on all the datasets. We observe that instruction tuning provides more performance gains using MoLA compared with LoRA. For example, our MoLA-▽ outperforms LoRA by 0.3 on MRPC in the direct fine-tuning setting, and this improvement increases to 1.04 in this setting. With instruction tuning, MoLA-▽ achieves either equal or better performance compared with MoLA-□(8888) on all the datasets even with much fewer parameters.

Tables 1 and 2 show that MoLA-△ and -⋈ perform worse than MoLA-□ and MoLA-▽, especially in the QA task. Of all MoLA variants, MoLA-▽ generally achieves the best performance, outperforming all other variants on five benchmarks. We performed the Wilcoxon signed-rank test with False Discovery Rate (FDR) correction among MoLA-▽ and other baselines (Prompt Tuning, LLaMA-Adapter, LoRA) based on their accuracy on six benchmarks. The superior performance of MoLA-▽ compared to other baselines was verified to be statistically significant with all p-values being less than 0.05.

These experiments show that allocating more experts in the higher layers and fewer experts in the lower layers provides better effectiveness compared with other allocation strategies. We therefore argue that the number of experts at the top layers is important. In other words, if we would like to prune the MoLAs, it is better to reduce the number of experts at lower layers to reduce the trainable parameters. In the next section, we explore the possible rationales behind this configuration and some other properties of MoLA.

5. Model Analysis and Ablation Studies

5.1. Analysis of Layer-wise Experts Redundancy

In the previous section, we observed that allocating more LoRA experts to higher layers provides more performance gains. Thus, based on our assumptions, higher layers should be allocated more experts than the lower layers. More experts with fine-grained processing on token-level information may introduce redundancy. On the contrary, higher layers require more experts because higher layers learn more abstract and high-level information. More experts can enhance the architecture to learn fine-grained and task-specific patterns for complicated downstream problems. Here, we study the redundancy of the layer-wise LoRA experts to convince our assumptions.

To better analyze the models, we formally define the layer-wise expert redundancy as follows:

Definition 5.1. Expert Redundancy measures the layer-wise difference between expert modules in MoE architecture for Transformer models.

When two selected experts are similar, they may overlap and create some redundancy. To quantitatively examine the Expert Redundancy of Transformer layer j, we calculate the average value of the Frobenius Norm between any two different LoRA experts' weight matrices in each self-attention module from layer j. Figure. 3 presents the layer-wise average values of MoLA-□(8888) and MoLA-□(5555), where both models are trained via the sampled instruction-tuning dataset. In the figure, the average value of the Frobenius Norm per layer increases from lower layers to higher layers, showing that experts in lower layers are more similar than those in higher layers. We also find a similar pattern from other MoLA configurations, as demonstrated in Appendix. B. This observation supports our assumption that experts in lower layers of the Transformer model suffer more expert redundancy. Therefore, within a certain number of experts in total, allocating more experts in higher layers is more effective in improving the model performance.

We also analyze the average router weights and selected times of experts in each layer. Analysis shows that most of the experts are selected for a similar workload and utilized sufficiently (details are described in Appendix. C). With similar selected weights and times, lower layers have more expert redundancy due to more similar experts, again supporting our statement.

5.2. Continuous Learning

MoE-based architectures leverage the sparse sub-modules to process information, and thus only selected modules are optimized for different input types. This feature may also provide more stable performance for continuous learning. Here, we explore the domain continuous learning ability of our MoLAs and perform experiments on the ScienceQA dataset. We choose the 5 topics with the most training samples including biology, physics, chemistry, economics, and earth science. We continuously fine-tune MoLAs on new domains and study the performance drop on previous domains. Following Chaudhry et al. (2018), we calculate the overall performance (OP):

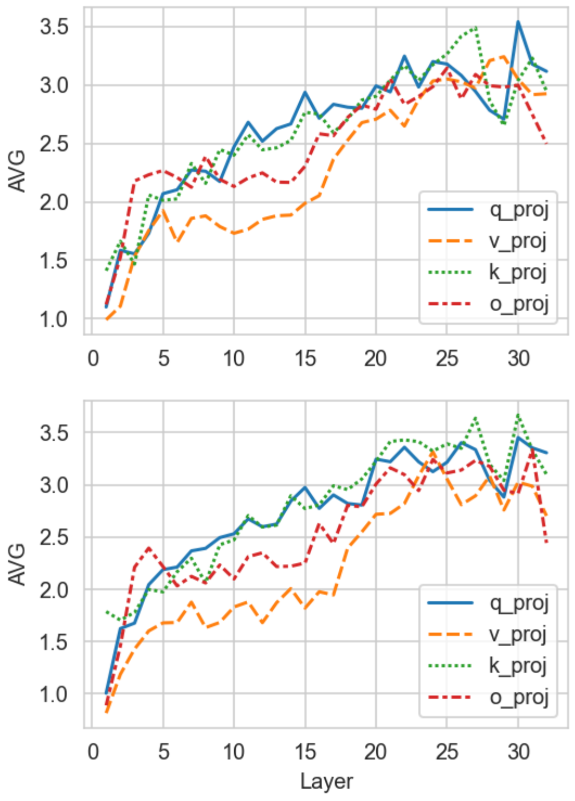

Figure 3. Average number of the Frobenius Norm between two different experts' weight matrices for each self-attention module from each layer. The top figure is for the MoLA-□(8888), and the bottom figure is for MoLA-□ (5555). Both models are trained via instruction tuning.

$$OP = \frac{1}{t}\sum_{i=1}^{t} R_{t,i}, \tag{5}$$

where t is the number of domain and $R_{t,i}$ denotes the model's accuracy on domain i after continuously trained on domain t. We also propose a performance drop score to measure the domain forgetting by calculating the performance drop in the continuous learning process, as illustrated in the following equation:

$$PD = \frac{1}{t(t-1)/2} \sum_{k=2}^{t} \sum_{i=1}^{k-1} (R_{k,i} - R_{k-1,i}), \tag{6}$$

As shown in Table 3, MoLAs can achieve better overall performance than LoRA. Specifically, MoLA-▽ shows the superior ability to avoid domain knowledge forgetting by having a -0.47 performance drop score, which aligns with our insights that the higher layers have less expert redundancy. The detailed results are shown in Appendix. A.

Table 3. Comparison with various MoLA in a continuous setting.

Models	OP ↑	PD ↑
LoRA	78.67%	-2.17%
MoLA-□ (5555)	88.80%	-0.6%
MoLA-⋈ (8228)	83.82%	-3.92%
MoLA-△ (8642)	88.84%	-2.10%
MoLA-▽ (2468)	**89.82%**	**-0.47%**

6. Related Work

Here, we describe works most closely linked to this effort. These include work on parameter-efficient tuning and recent research combining MoE and parameter-efficient tuning.

6.1. Parameter-Efficient Tuning

Parameter-efficient tuning of LLMs has garnered considerable attention because it is cost-effective nature for fine-tuning LLMs. Li & Liang (2021) and Liu et al. (2022b) present the use of soft prompting concatenated to either the embedding layer or intermediate layers of the Transformer model. However, these approaches involve adding extra embedding tokens to the sequence, potentially compromising efficiency during inference, especially in the case of long input contexts. Hu et al. (2022) introduces the LoRA parameter-efficient adaptation technique which uses low-rank decomposition matrices of dense weight matrices of Transformers. LoRA achieves decent performance for fine-tuning LLMs without additional inference costs. Similarly, Liu et al. (2022a) uses task-specific vectors to modify attention activation, also avoiding extra inference costs. Inspired by these approaches, our approach combines the MoE technique with parameter-efficient tuning approaches and leverages the layer-wise expert allocation to further push the limit of performance.

6.2. Parameter-Efficient MoE

Some recent efforts have studied the integration of MoE and parameter-efficient tuning methods to improve the effectiveness of instruction tuning. Liu et al. (2023) applies MoE with LoRA matrices for fine-tuning language models on various medical domain tasks. This method takes the task type as an additional input for training the router, which requires additional prior knowledge during inference. Our approach does not require additional prior knowledge since our MoLA experts are learned without supervision. Dou et al. (2023) introduced LoRAMoE, a novel adapter architecture that combines MoE and LoRA within the feed-forward layer of each Transformer block. This effort also studies how to mitigate knowledge forgetting in LLMs during traditional supervised fine-tuning. However, this paper only applies LoRAMoE on the feed-forward layer in each Transformer block. MoLA, on the other hand, applies LoRA experts across each dense weight matrix in the Transformer, further improving both the performance and scalability of parameter-efficient fine-tuning. Zadouri et al. (2023) introduces a framework that combines MoE with various parameter-efficient architectures, including LoRA and IA3 (Liu et al., 2022a), called MoLORA and MoV. Their experiments show that their framework leverages instruction tuning more effectively than prior parameter-efficient architectures, improving the zero-shot capabilities of LLMs. However, they did not study how this framework works on decoder-only LLMs and task-specific fine-tuning. Furthermore, the previously mentioned methods do not consider the layer-wise allocation of experts. Our MoLA approach introduces a novel design that allows for a varying number of experts in each layer, therefore further improving the effectiveness of LoRA-MoE approaches.

7. Conclusion

We introduce MoLA, a novel parameter-efficient tuning approach that leverages layer-wise expert allocation in the MoE and combines it with the LoRA technique. We propose four layer-wise expert configurations, MoLA-△, MoLA-▽, MoLA-⋈, and MoLA-□ based on different assumptions. Our comprehensive experiments on six popular benchmarks including NLP and commonsense question-answering tasks demonstrate that MoLA significantly outperforms other baselines. Specifically, MoLA-▽ achieves the best performance in all the configurations, convincing our assumption that with a certain number of experts in total, higher layers need to be allocated more experts. We conduct extensive analysis to explore the layer-wise expert redundancy, observing that lower layers of the Transformer model suffer higher expert redundancy with MoLA tuning. Ablation studies also show that MoLA has promising continuous learning capability. As a plug-and-play PEFT approach, MoLA can be used on wide tasks. Furthermore, this work provides a promising research direction to enhance the MoE technique and PEFT approach. In the future, we will explore dynamic learning layer-wise expert allocation and apply this approach to more diverse tasks.

Broader Impact

This paper contributes to the advancement of parameter-efficient tuning within the field of machine learning. To the best of our knowledge, it does not directly raise any specific ethical concerns. However, it is important to note that our research relies on pre-trained large language models that may exhibit preferential biases (Tang et al., 2023). Users of these models should be cognizant of these biases and consider their potential implications. Our approach is a plug-and-play parameter-efficient tuning method and can be used for diverse tasks. We push the performance limits of PEFT methods and provide decent performance on LLM fine-tuning while dramatically reducing training costs. We believe that this research direction will benefit energy saving and advance the decarbonization of AI. Also, efficient training efficiency and promising performance promote wider groups of people to leverage our approach on more practical problems.

References

Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A., Hauth, A., et al. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Chaudhry, A., Dokania, P. K., Ajanthan, T., and Torr, P. H. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 532–547, 2018.

Chen, T., Zhang, Z., Jaiwal, A., Liu, S., and Wang, Z. Sparse moe as the new dropout: Scaling dense and self-slimmable transformer. *ICLR*, 2023.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

Dolan, W. and Brockett, C. Automatically constructing a corpus of sentential paraphrases. *IJCNLP*, 2005.

Dou, S., Zhou, E., Liu, Y., Gao, S., Zhao, J., Shen, W., Zhou, Y., Xi, Z., Wang, X., Fan, X., Pu, S., Zhu, J., Zheng, R., Gui, T., Zhang, Q., and Huang, X. Loramoe: Revolutionizing mixture of experts for maintaining world knowledge in language model alignment. *arXiv preprint arXiv:2312.09979*, 2023.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270, 2022.

Hu, E., Shen, Y., Wallis, P., Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. *ICLR*, 2022.

Jiang, A., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D., Casas, D. d. l., et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Li, X. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. *ACL*, 2021.

Lian, W., Goodson, B., Pentland, E., Cook, A., Vong, C., and ”Teknium”. Openorca: An open dataset of gpt augmented flan reasoning traces. `https://https://huggingface.co/Open-Orca/OpenOrca`, 2023.

Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *NeurIPS*, 2022a.

Liu, Q., Wu, X., Zhao, X., Zhu, Y., Xu, D., Tian, F., and Zheng, Y. Moelora: An moe-based parameter efficient fine-tuning method for multi-task medical applications. *arXiv preprint arXiv:2310.18339*, 2023.

Liu, X., Ji, K., Fu, Y., Tam, W., Du, Z., Yang, Z., and Tang, J. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. *ACL 2022*, 2022b.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Lu, P., Mishra, S., Xia, T., Qiu, L., Chang, K.-w., Zhu, S.-C., Tafjord, O., Clark, P., and Kalyan, A. Learn to explain: Multimodal reasoning via thought chains for science question answering. *NeurIPS*, 2022.

Lv, K., Yang, Y., Liu, T., Gao, Q., Guo, Q., and Qiu, X. Full parameter fine-tuning for large language models with limited resources. *arXiv preprint arXiv:2306.09782*, 2023.

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? a new dataset for open book question answering. *EMNLP*, 2018.

Mishra, S., Khashabi, D., Baral, C., and Hajishirzi, H. Cross-task generalization via natural language crowdsourcing instructions. In *60th Annual Meeting of the Association* *for Computational Linguistics, ACL 2022*, pp. 3470–3487. Association for Computational Linguistics (ACL), 2022.

Mukherjee, S., Mitra, A., Jawahar, G., Agarwal, S., Palangi, H., and Awadallah, A. Orca: Progressive learning from complex explanation traces of gpt-4, 2023.

Sanh, V., Webson, A., Raffel, C., Bach, S. H., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Le Scao, T., Raja, A., et al. Multitask prompted training enables zero-shot task generalization. In *ICLR 2022-Tenth International Conference on Learning Representations*, 2022.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *ICLR*, 2017.

Singhal, K., Azizi, S., Tu, T., Mahdavi, S., Wei, J., Chung, H., Scales, N., Tanwani, A., et al. Large language models encode clinical knowledge. *arXiv preprint arXiv:2212.13138*, 2022.

Talmor, A., Herzig, J., Lourie, N., and Berant, J. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *NAACL*, 2019.

Tang, R., Zhang, X., Lin, J., and Ture, F. What do llamas really think? revealing preference biases in language model representations. *arXiv preprint arXiv:2311.18812*, 2023.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. Glue: A multi-task benchmark and analysis platform for natural language understanding. *ICLR*, 2019.

Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2021.

Zadouri, T., Ustun, A., Ahmadian, A., Ermis, B., Locatelli, A., and Hooker, S. Pushing mixture of experts to the limit: extremely parameter efficient moe for instruction tuning. *arXiv preprint arXiv:2309.05444*, 2023.

Zhang, Q., Chen, M., Bukharin, A., He, P., Cheng, Y., Chen, W., and Zhao, T. Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023a.

Zhang, R., Han, J., Zhou, A., Hu, X., Yan, S., Lu, P., Li, H., Gao, P., and Qiao, Y. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*, 2023b.

Zhu, L., Wang, X., and Wang, X. Judgelm: Fine-tuned large language models are scalable judges. *arXiv preprint arXiv:2310.17631*, 2023.

Zoph, B., Bello, I., Kumar, S., Du, N., Huang, Y., Dean, J., Shazeer, N., and Fedus, W. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202:08906*, 2022.

A. Continuous Learning

We evaluate the performance of LoRA and our MoLAs in the domain's continuous learning setting. We fine-tuning the models on biology, physics, chemistry, economics, and earth-science domains sequentially. The training epochs are 20, and we use the same hyper-parameters as we used for direct fine-tuning. The detailed results are shown in Table. 4, where the score of Bio-Phy denotes the result when the model is trained on the biology domain and tested on the physics domain.

Table 4. The results in the continuous learning setting. Bio-Phy denotes that the model trained on the biology domain is tested on the physics domain.

	LoRA	MoLA-$\triangle$ (5555)	MoLA-$\bowtie$ (8228)	MoLA-$\triangledown$ (8642)	MoLA-$\triangledown$ (2468)
Bio-Bio	92.19%	94.71%	95.97%	95.97%	94.96%
Bio-Phy	61.46%	60.94%	64.06%	64.58%	66.67%
Bio-Chem	55.46%	63.87%	55.46%	57.98%	61.34%
Bio-Econ	60.71%	72.62%	66.67%	70.24%	66.67%
Bio-Earth	52.31%	63.08%	55.38%	52.31%	61.54%
Phy-Bio	88.16%	91.44%	91.44%	92.44%	91.69%
Phy-Phy	89.58%	89.06%	92.19%	90.10%	88.54%
Phy-Chem	55.46%	36.13%	51.26%	57.14%	52.10%
Phy-Econ	72.62%	79.76%	78.57%	80.95%	77.38%
Phy-Earth	61.54%	58.46%	76.92%	66.15%	63.08%
Chem-Bio	85.14%	90.43%	82.62%	89.67%	87.91%
Chem-Phy	81.77%	81.25%	85.42%	85.42%	91.15%
Chem-Chem	93.28%	94.96%	94.96%	95.80%	94.12%
Chem-Econ	59.52%	61.90%	58.33%	48.81%	72.62%
Chem-Earth	58.46%	60.00%	60.00%	52.31%	66.15%
Econ-Bio	78.84%	89.67%	83.12%	88.66%	87.15%
Econ-Phy	78.13%	88.02%	83.85%	87.50%	86.98%
Econ-Chem	70.59%	92.44%	92.44%	96.64%	94.12%
Econ-Econ	73.81%	82.14%	94.05%	94.05%	86.90%
Econ-Earth	38.46%	56.92%	60.00%	47.69%	61.54%
Earth-Bio	84.13%	88.66%	83.38%	86.15%	88.16%
Earth-Phy	77.08%	85.94%	81.77%	85.42%	91.67%
Earth-Chem	87.39%	93.28%	89.08%	94.12%	90.76%
Earth-Econ	78.57%	86.90%	83.33%	89.29%	89.29%
Earth-Earth	66.15%	89.23%	81.54%	89.23%	89.23%

B. Frobenius Norms of Different Experts Allocation

We show the Frobenius Norms for various MoLA in Figure. 4. In the figure, the top sub-figure is for the MoLA-$\triangledown$ with configuration as 2468; the middle sub-figure is for the MoLA-$\triangle$ with configuration as 8642; and the bottom sub-figure is for MoLA-$\bowtie$ with configuration 8228 after instruction tuning. All various MoLAs follow the same pattern, and the difference between the weight matrices of the experts becomes larger as the layer becomes higher.

C. Analysis of Average Fusion Weights and Selected Times of Experts

We also calculate the average fusion weights provided by the router and average times for experts when they are selected, as shown in Figure 5. In Figure 5 (a), we find that most fusion weights are around 0.5, which means the importance of selected experts is similar most of the time. Furthermore, in Figure 5 (b), although there are several experts are not often selected, most of the experts are selected frequently and utilized sufficiently, which supports our insights as well.

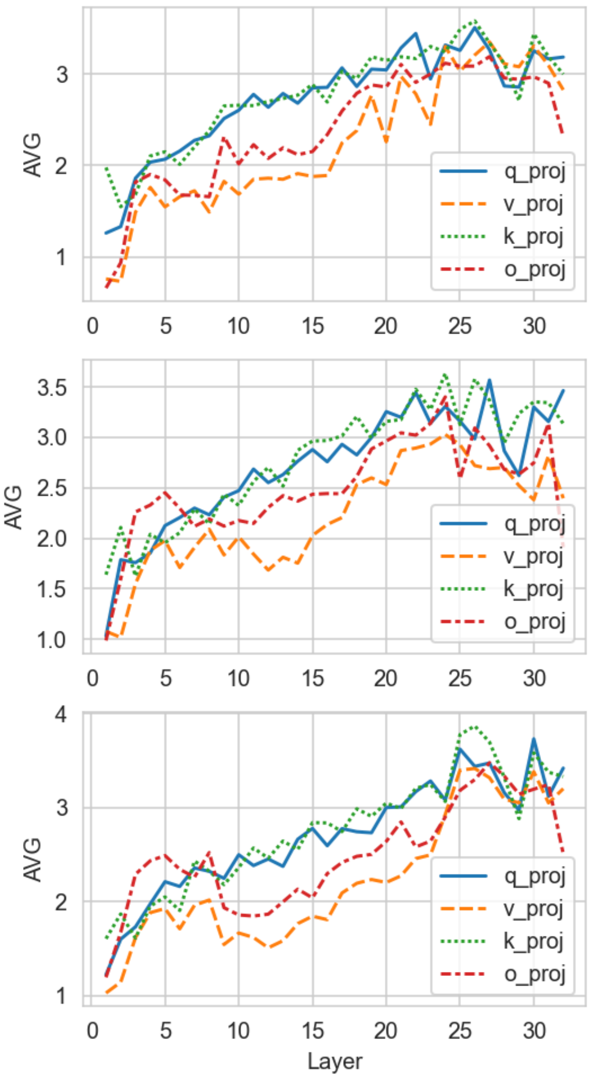

Figure 4. The average number of the Frobenius Norm between two different experts' weight matrices at the same layer for each self-attention module. The top figure is for the MoLA-▽ with configuration as 2468; the middle figure is for the MoLA-△ with configuration as 8642; and the bottom figure is for MoLA-⋈ with configuration 8228 after instruction tuning.

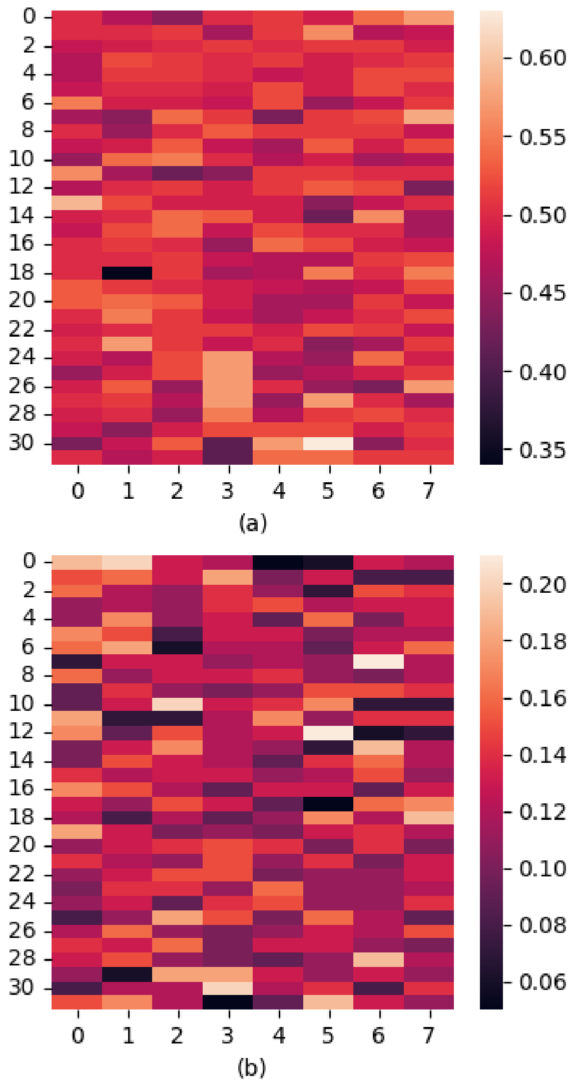

Figure 5. (a) The average fusion weights for each expert. (b) The average times for each expert when it is selected.