EM-LoRA: Efficient Mixture of Low-Rank Adaptation for Large Language Models Fine-tuning

Anonymous ACL submission

Abstract

Low-rank adaptation (LoRA) and its mixture-of-experts (MOE) variants are highly effective parameter-efficient fine-tuning (PEFT) methods. However, they introduce significant latency in multi-tenant settings due to the LoRA modules and MOE routers added to multiple linear modules in the Transformer layer. To address this issue, we propose the Efficient Mixture of Low-Rank Adaptation (EM-LoRA), a novel LoRA variant. EM-LoRA differs from previous MOE-style LoRA methods by considering each LoRA module as an expert and employing a prompt-aware routing mechanism. This mechanism calculates expert routing results once before generating the first new token and reuses these results for subsequent tokens, reducing latency. Extensive experiments and analysis on commonsense reasoning tasks, math reasoning tasks, and widely used LLM evaluation benchmarks demonstrate that EM-LoRA consistently outperforms strong PEFT baselines with comparable tunable parameter budgets. Additionally, EM-LoRA significantly reduces latency in multi-tenant settings compared to previous LoRA-based methods.[1]

1 Introduction

Large language models (LLMs) have been achieving state-of-the-art (SOTA) results not only in various natural language processing tasks (Qin et al., 2023; Zhu et al., 2023) but also in numerous challenging evaluation tasks (Huang et al., 2023; Li et al., 2023), such as question answering, reasoning, math, safety, and instruction following. Although LLMs are evolving into general task solvers, fine-tuning remains essential for efficient LLM inference and for controlling the style of the generated content (Xin et al., 2024; Ding et al., 2022). Full-parameter fine-tuning of such large models is impractical due to the significant GPU memory and computational resources required. Consequently, parameter-efficient fine-tuning (PEFT) (Zhang et al., 2023b; Zhao et al., 2023) has garnered considerable attention in the research community, as it typically involves tuning less than 1% of the LLMs' parameters, thereby substantially reducing computational costs.

Among many PEFT methods, the reparameterization-based method low-rank adaptation (LoRA) (Hu et al., 2021) is considered one of the most effective methods for LLMs (Xu et al., 2023; Ding et al., 2022; Xin et al., 2024). Although LoRA is effective and can bring stable performance with the original setting in Hu et al. (2021), it still brings inconvenience under the multi-tenant setting (Chen et al., 2023): it has to add LoRA modules to multiple weights of the Transformer layer and introducing significant additional latency in every generation steps under the multi-tenant setting. Recently, the Mixture-of-Experts (MOE) style LoRA methods (Chen et al., 2024; Yang et al., 2024; Liu et al., 2023; Dou et al., 2023; Gou et al., 2023) have surged, further pushing the performance ceilings of LoRA fine-tuning. However, they introduce the calculation of MOE routers, further increasing inference latency. Thus, it is essential to develop a novel variant of the LoRA method that introduces minimum latency during generation and still can perform competitively in downstream tasks.

In this work, we propose a novel PEFT method called Efficient Mixture of Low-Rank Adaptation (EM-LoRA). Our EM-LoRA method differs from the previous literature on MOE-style LoRA methods in the following two aspects. First, in EM-LoRA, an entire LoRA module is considered a LoRA expert, and the LoRA router is responsible for determining which LoRA expert to activate. Second, we propose the prompt-aware routing mechanism instead of calculating the expert

[1] Codes and fine-tuned models will be open-sourced to facilitate future research.

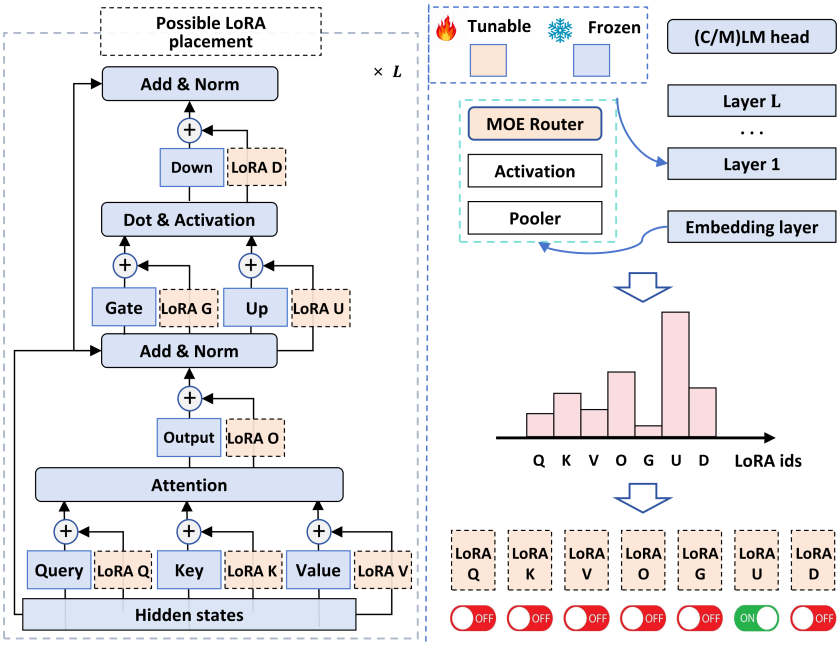


Figure 1: Schematic illustration of our EM-LoRA method. **Left**: The architecture of a Transformer layer as in LlaMA-2 (Touvron et al., 2023). There are seven linear modules and seven positions to add LoRA modules. **Right**: Upon receiving an input prompt, the LoRA router before each Transformer layer will take the input prompt's hidden states as input features and go through a pooler, an activation function, and the MOE router network to determine which LoRA module is activated (or used) (e.g., LoRA U in the figure). This routing decision is repeatedly used when generating subsequent tokens.

routing results for every new token. Given an input prompt, the expert routing results are calculated once, right before the generation of the first new token. The subsequent generation steps will reuse the expert routing results. Under the prompt-aware routing mechanism, our LoRA router consists of a pooler operation, a learnable activation function (Molina et al., 2019), and a sparse MOE router.

We conduct extensive experiments and analysis on various challenging tasks, including five commonsense reasoning tasks, two math reasoning tasks, and three widely used LLM evaluation benchmarks. Our method can consistently outperform strong PEFT baselines with comparable tunable parameter budgets, especially the recent LoRA variants. In addition, our EM-LoRA method has significantly lower latency under the multi-tenant setting (Chen et al., 2023) than the previous LoRA-based methods with comparable tunable parameters.

Our contributions are summarized as follows:

- we propose a novel LoRA variant, EM-LoRA, which combines the MOE mechanism with LoRA in an efficient way.

- In EM-LoRA, we treat each LoRA module as an expert.

- We propose a prompt-aware routing mechanism to avoid token-wise router calculations.

- We have conducted extensive experiments and analysis showing that our EM-LoRA framework is (a) practical and outperforms the baselines under comparable parameter budgets. (b) efficient during inference for LLMs.

2 Related works

Due to limited length, we put more related works in parameter-efficient fine-tuning to Appendix A.

2.1 The LoRA method and its variants

Since LoRA is the most popular PEFT method in the era of large language models, many works are devoted to improving upon LoRA. AdaLoRA (Zhang et al., 2023a) looks into the parameter allocation of LoRA modules. VERA (Kopiczko et al., 2023) investigate whether one could freeze the randomly initialized LoRA matrices and only learn a set of scaling vectors. Recently, a series of works has been looking into combining Mixture-of-Experts (MoE) (Shazeer et al., 2017; Jacobs et al., 1991) and LoRA. LLaVA-MoLE (Chen et al., 2024) effectively routes tokens to domain-specific LoRA experts, mitigating data conflicts and achieving consistent performance gains over the original LoRA method. MOELoRA (Liu et al., 2023) proves that fine-tuning LoRA modules with a MOE router enables the LLMs to perform well in a multi-task learning setting. MoRAL (Yang et al., 2024) addresses the challenge of adapting LLMs to new domains/tasks and enabling them to be efficient lifelong learners using the MOE techniques. LoRAMoE (Dou et al., 2023) integrates LoRAs using a router network to alleviate world knowledge forgetting after instruction tuning. MoCLE (Gou et al., 2023) proposes a MoE architecture to activate task-customized model parameters based on instruction clusters.

Although performing well in fine-tuning, these methods introduce high additional latency since (a) these methods do not reduce the number of LoRA modules in the Transformer backbone. (b) the routers and LoRA modules must be called when generating each new token. Our EM-LoRA method addresses this efficiency issue by (a) only calling the LoRA routers when encoding the input prompt and before generating the first new token. (b) only activate one LoRA module per Transformer layer.

3 Methods

In this section, we first introduce the foundational concepts of LoRA and MoEs and then elaborate on the architectural design of EM-LoRA.

3.1 Preliminaries

Transformer model As depicted in Figure 1, each Transformer layer of a LLM such as LlaMA-2 (Touvron et al., 2023) consists of a multi-head self-attention (MHA) sub-layer and a fully connected feed-forward (FFN) sub-layer. MHA contains four linear modules, which are the Query (Q), Key (K), Value (V), and Output (O) modules. FFN contains three linear modules: Gate (G), Up (U), and Down (D). For notation convenience, we will refer to the number of modules in a Transformer block as N_{mod}. Thus, in LlaMA-2, $N_{mod} = 7$.

LoRA For any Transformer module $m \in$ {Q, K, V, O, G, U, D}, the LoRA method adds a pair of low-rank matrices to reparameterize its weights. Formally, the forward calculation of module m with LoRA is:

$$x^{'} = xW_m + xW_m^A W_m^B + b_m, \qquad (1)$$

where $W_m \in \mathbf{R}^{d_1 \times d_2}$ is the weight matrix of module m, b_m is its bias term. $W_m^A \in \mathbb{R}^{d_1 \times r}$ and $W_m^B \in \mathbb{R}^{r \times d_2}$ are the low-rank matrices for the LoRA module, and $r \ll \min(d_1, d_2)$. r is the rank of the two matrices and will also be referred to as the rank of the LoRA module.

3.2 Motivation

As demonstrated later in Table 3, the existing works on MOE style LoRA significantly slow down the LLM backbone during inference, reducing tokens per second (tps) by around 20%

RQ1. *Can we treat a LoRA module as an expert so that each Transformer layer has only one LoRA router and activate only one such expert per layer?*

RQ2. *Can the LoRA router be called once for an input prompt?*

3.3 Prompt-aware LoRA router

Trying to investigate ***RQ1*** and ***RQ2***, we now try to propose the details of our EM-LoRA method. The core of EM-LoRA is the prompt-aware routing mechanism. Under this mechanism, the LoRA router takes the input prompt's hidden states as input and outputs the activated LoRA experts for the current layer. Different from the previous works (Chen et al., 2024; Yang et al., 2024; Liu et al., 2023; Dou et al., 2023; Gou et al., 2023), our work: (a) only calculates the LoRA routers once when the input prompt is fed through the Transformer backbone for the first time and right before generating the first new token. The routers' activation decisions will be repeatedly used in the subsequent generation steps. (b) determine the activated LoRA experts at the Transformer's layer level, selecting which Transformer module is modified by its corresponding LoRA module.

As shown in Figure 1, to generate a response, the input prompt has to go through the LLM backbone to obtain the hidden representations. Denote the hidden state of the input prompt with length n_p right before Transformer layer l as $\mathbf{H}^l \in \mathbf{R}^{n_p \times d}$. Then a pooling operation Pooler() aggregates the semantic information in $\mathbf{H}^l$ and transforms it to $\mathbf{h}^l \in \mathbf{R}^{1 \times d}$:

$$\mathbf{h}^l = \text{Pooler}(\mathbf{H}^l). \qquad (2)$$

Here, according to (Zhu, 2021b,a), the Pooler operation can be one of the following: (a) last-token pooling, which is to use the vector representation of the last token in the prompt as $\mathbf{h}^l$. This pooler is widely used when decoder-based models perform sentence classification tasks. (b) average pooling.

(c) max pooling. (d) self-attention-based pooling, whose detail is introduced in Appendix D.

Then, $\mathbf{h}^l$ will go through an activation function g and then the LoRA router R^l right before layer l. R^l assigns the current input prompt to the most suitable LoRA expert. This router contains (a) a linear layer that computes the probability of $\mathbf{h}^l$ being routed to each LoRA expert LoRA$_m$, (b) a softmax function to model a probability distribution over the LoRA experts, and finally, (c) a Top-k function that choose the top $k > 0$ experts with the highest probability masses. Formally,

$$R^l(\mathbf{h}^l) = \text{Top-k}(\text{Softmax}(g(\mathbf{h}^l)W_r^l)), \quad (3)$$

where $W_r^l \in \mathbf{R}^{d \times N_{mod}}$ is the router's weight. The LoRA router dynamically selects the best k experts for each input prompt during inference. Note that the router is only called once before a new token is generated. The activated LoRA experts are used throughout the whole generation process.

Following Fedus et al. (2022), we add a load balancing loss to the training loss function. Consider a training batch B with N_B samples, let f_i^l represent the proportion of prompts assigned to the i-th LoRA expert in layer l,

$$f_i^l = \frac{1}{N_B} \sum_{x \in B} \mathbf{1}\{\arg\max_j p_j^l(x) = i\}, \quad (4)$$

where p_j^l is the probability of expert j, output by the router l. Let $\hat{p}_i^l$ be the average of probability masses received by the i-th expert, $\hat{p}_i^l = \frac{1}{N_B} \sum_{x \in B} p_i^l(x)$. Then, the load balancing loss is given by:

$$\mathcal{L}_{lb} = N_{mod} \sum_{i=1}^{N_{mod}} f_i^l \cdot \hat{p}_i^l. \quad (5)$$

The $\mathcal{L}_{lb}$ loss term is added to the cross entropy loss with a coefficient $\lambda_{lb} \geq 0$.

3.4 Learned activation functions

The previous PEFT literature usually set the activation functions in a PEFT module to be ReLU (Mahabadi et al., 2021; Pfeiffer et al., 2021; Liu et al., 2022b) and does not discuss whether this setting is optimal. In addition, the PEFT modules' activation functions in different Transformer layers are usually set to be identical. As will be presented later in Table 4, it is beneficial for LoRA routers of different depths to have different activation functions. Thus, how can we find an optimal setting for the LoRA routers' activation functions? Exhaustive hyper-parameter search is time and GPU-consuming. Thus, we are motivated to set the activation function to be learnable during training.

We resort to rational activation functions (Molina et al., 2019), which are learnable and can approximate common activation functions and learn new ones. The rational activation function $R(x)$ of order m, n is defined as follows:

$$\text{Ra}(x) = \frac{\sum_{j=0}^{m} a_j x^j}{1 + \| \sum_{i=1}^{n} b_i x^i \|}, \quad (6)$$

where a_j and b_i are learnable parameters. The rational activation functions are successfully applied in image classification (Molina et al., 2019) and sequence modeling (Delfosse et al., 2021).

Inspired by the above literature, we propose learning the activation functions in LoRA routers via the rational activation functions when finetuning a downstream task. Denote the set of parameters in the learnable activations as Θ and the other parameters in the LoRA routers and LoRA experts as Ω. Following DARTS (Liu et al., 2019), we consider Θ as architectural parameters and optimize them along with Ω via bi-level optimization. Due to limited length, we introduce bi-level optimization in Appendix B.

4 Experiments

In this section, we conduct a series of experiments and analysis to evaluate our EM-LoRA method.

4.1 Datasets and evaluation metrics

We compare our approach to the baselines on a collection of challenging tasks: (a) five benchmark common-sense question-answering tasks, ARC-e and ARC-c (Clark et al., 2018), OBQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), BoolQ (Clark et al., 2019). (b) two math reasoning tasks, AQuA (Ling et al., 2017) and GSM8k (Cobbe et al., 2021). We utilize the chain-of-thought (COT) rationales for these samples provided by Hu et al. (2023) for training on these math tasks. All rationales are generated through zero-shot CoT (Wei et al., 2022; Kojima et al., 2022) on GPT-3.5[2], but without undergoing any error filtering. (c) MT-Bench (Zheng et al., 2023), MMLU (Hendrycks et al., 2020), and BBH (Suzgun et al., 2022). Since

[2] https://platform.openai.com/docs/models

these tasks provide no training data, we utilize the Alpaca (Taori et al., 2023) dataset for instruction tuning. More detailed introduction, statistics, and evaluation metrics can be found in Appendix C.

4.2 Baselines

We compare our EM-LoRA framework with the current SOTA PEFT baseline methods.

LoRA and its variants we consider the following LoRA variants as baselines: (a) the original LoRA (Hu et al., 2021); (b) AdaLoRA (Zhang et al., 2023a), which adaptively adjust the LoRA parameters among different Transformer modules. (c) MOELoRA (Liu et al., 2023), which considers each LoRA module as a mixture of single-rank LoRA experts. (d) DoRA (Liu et al., 2024), one of the most recent variants of LoRA that decomposes the pre-trained weights into two components, magnitude, and direction, for fine-tuning, specifically employing LoRA for directional updates.

Other PEFT methods We also consider the most recent PEFT methods: (a) Parallel-Adapter proposed by He et al. (2021); (b) Learned-Adapter (Zhang et al., 2023b). (c) P-tuning v2 (Liu et al., 2021). (d) IAPT (Zhu et al., 2024). (e) BitFit (Ben-Zaken et al., 2021). (f) $(IA)^3$ (Liu et al., 2022a), which multiplies learnable vectors to the hidden states in different modules of the Transformer layer. (g) SSP (Hu et al., 2022), which is a representative work on combining different PEFT methods, including LoRA and BitFit.

The baselines are implemented using their open-sourced codes. We only adjust the hyper-parameters related to tunable parameter numbers to fairly compare the baseline methods and our EM-LoRA method. The hyper-parameter settings for the baselines are detailed in Appendix F.

4.3 Experiment Settings

Computing infrastures We run all our experiments on NVIDIA A40 (48GB) GPUs.

Pretrained backbones The main experiments use the most recent open-sourced LLMs, LlaMA-2 7B (Touvron et al., 2023) as the pretrained backbone model. In the ablation studies, we will also use the recently released LlaMA-2 13B and Gemma 2B (Team et al., 2024).

Prediction heads When fine-tuning LlaMA-2 7B, we only consider the supervised fine-tuning (SFT) setting (Ouyang et al., 2022). After receiving a prompt or instruction, all the predictions are generated using the language modeling head (LM head). No additional prediction heads are installed to make categorical or numerical predictions. For decoding during inference, we use beam search with beam size 3.

Hyper-parameters for the EM-LoRA framework In our experiments, unless otherwise specified, we set: (a) the rank of each LoRA expert is set to $r = 32$. (b) k is set to 1. That is, each router activates one expert. (c) the LoRA router adopts the self-attention pooler. (d) the hyper-parameters of the rational activation are $m = 6$, $n = 5$, and th e learnable parameters a_j and b_i are initialized by approximating the GeLU activation function. (e) λ_{lb} is set to 1e-2. Under the above settings, our EM-LoRA method will introduce 80.9M tunable parameters and, at most, 16.4M activated PEFT parameters to the LlaMA-2 7B backbone. The hyper-parameters for training are specified in Appendix F.

Reproducibility We run each task under five different random seeds and report the median performance on the test set of each task.

Due to limited length, other experimental settings for the baseline methods and the training procedure are in Appendix F.

4.4 Main results

Single-task setup. In this setup, We compare EM-LoRA with baseline PEFT methods by employing these methods for fine-tuning a single task. The experimental results on the five commonsense reasoning tasks and two math reasoning tasks are presented in Table 1. We present the number of tunable parameters in the second column and the average activated parameters in the third column. Table 1 reveals that our EM-LoRA method outperforms the baseline methods across all seven tasks, with comparable tunable parameters and much fewer activated parameters. In particular, EM-LoRA outperforms the previous SOTA LoRA style baselines like AdaLoRA, DoRA, and MOELoRA with comparable parameters. These results demonstrate that our method is good at downstream task adaptation of large language models.

Multi-task setup. Table 6 of Appendix G presents the results of LoRA, DoRA, MOELORA, and EM-LoRA with LLaMA2-7B in multi-task learning. In contrast to the single-task setup in Table 1, during multi-task learning, we mixed training data from ARC, BoolQ, OBQA, and PIQA to train the model, followed by separate evalua-

Method	Tunable Params	Activated Params	ARC-e (acc)	ARC-c (acc)	BoolQ (acc)	OBQA (acc)	PIQA (acc)	AQuA (acc)	GSM8k (acc)	Avg.
Baselines										
Parallel-Adapter	83.9M	83.9M	67.1	54.2	65.2	76.3	69.8	15.6	26.4	53.5
Learned-Adapter	81.8M	81.8M	69.3	54.4	64.9	78.4	75.6	18.3	28.9	55.7
P-tuning v2	84.5M	84.5M	63.5	51.3	61.2	76.1	66.2	9.63	21.1	49.9
IAPT	83.9M	83.9M	66.3	54.7	67.8	79.2	77.3	13.6	25.8	55.0
BitFit	87.0M	87.0M	65.9	54.1	66.4	77.2	76.6	11.8	21.7	53.4
$(IA)^3$	78.6M	78.6M	68.1	54.6	67.2	78.1	75.4	13.2	23.4	54.3
SSP	80.6M	80.6M	71.6	57.6	69.6	79.5	79.7	15.9	31.8	58.0
LoRA	80.0M	80.0M	73.4	57.2	68.8	80.1	81.4	16.6	31.1	58.4
AdaLoRA	80.0M	80.0M	73.8	57.9	69.2	80.4	82.1	17.6	31.7	59.0
MOELoRA	87.3M	17.3M	76.8	60.2	71.4	81.1	82.4	18.3	32.3	60.4
DoRA	80.0M	80.0M	76.5	59.8	71.7	80.6	82.7	17.9	32.6	60.3
Our proposed methods										
EM-LoRA (ours)	80.9M	12.1M	**77.8**	<u>61.2</u>	<u>72.6</u>	**81.7**	**83.2**	**19.9**	<u>33.9</u>	**61.5**
EM-DoRA (ours)	80.9M	12.6M	<u>77.5</u>	**61.3**	**72.7**	<u>81.3</u>	<u>83.0</u>	<u>19.3</u>	**34.1**	<u>61.3</u>

Table 1: The Overall comparison of different PEFT methods for single-task learning. The backbone model is LlaMA-2 7B. We report the median accuracy over five random seeds. Bold and Underline indicate the best and the second-best results.

tions to investigate the generalization ability of each method. The results indicate that (a) compared to single-task learning, LoRA and DoRA exhibit degradation in average accuracy in multi-task learning (LoRA: -2.0%, DoRA: -2.25%). At the same time, MOELORA and EM-LoRA maintain nearly the same average accuracy. EM-LoRA presents nearly no performance loss regarding the average score.

Results for general-purpose instruction tuning. After the LlaMA-2 7B is fine-tuned on the Alpaca (Taori et al., 2023) dataset with our EM-LoRA method or the MOELoRA methods, we utilize the challenging benchmarks, MT-Bench (Zheng et al., 2023), MMLU (Hendrycks et al., 2020), and BBH (Suzgun et al., 2022), for evaluation. We report the average GPT-4 score (gpt4-score) on the MT-Bench. Table 2 presents the results. Consistent with the previous experiments (Table 1 and 6), our EM-LoRA method outperforms the MOELoRA methods on the three benchmarks, demonstrating that EM-LoRA is superior in enhancing the instruction tuning quality of large language models. A case study of answers generated by different methods is presented in Table 7 of Appendix J, showcasing that EM-LoRA leads to better instruction-tuned LLMs.

4.5 Ablation studies and further analysis

Analysis of the inference efficiency To demonstrate the inference efficiency of our EM-LoRA method, we now compare the GPU memory and decoding speed of EM-LoRA, DoRA, and MOELoRA under beam search with different beam sizes. In this experiment, LoRA parameters are not merged to the backbone to mimic the single-LLM multi-tenant setting (Chen et al., 2023). The detailed settings are presented in Appendix H. We present two metrics for measuring efficiency: (a) peak memory cost (in MiB). (b) tokens generated per second (tps). The results are presented in Table 3.

From Table 3, under beam sizes 1 and 3, the EM-LoRA method has a comparable memory cost

Method	MT-Bench gpt4-score (↑)	MMLU acc	BBH acc
MOELoRA	7.08	48.2	36.8
EM-LoRA	7.21	49.7	37.3

Table 2: Performance of general-purpose instruction tuning using the EM-LoRA and MOELoRA methods. The backbone model is LlaMA-2 7B. ↑ means the metric is higher the better.

Method	Beam size	Speed (tps)	Memory cost (MiB)
DoRA	1	36.5	13784
	3	29.6	15292
MOELoRA	1	35.9	13788
	3	28.4	15352
EM-LoRA	1	43.7	13784
	3	33.5	15300

Table 3: The memory and speed of LlaMA-2 7B for generating responses given an input instruction (Appendix H), with different PEFT methods.

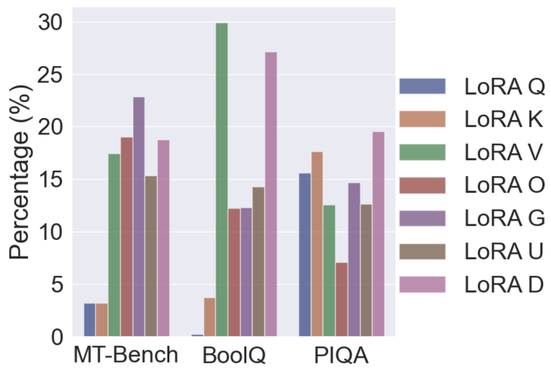


Figure 2: Distribution of LoRA experts across Transformer layers.

Method	BoolQ (acc)	PIQA (acc)	MMLU (acc)
EM-LoRA	**72.6**	**83.2**	**49.7**
EM-LoRA-1	72.4	83.1	49.5
EM-LoRA-2	72.2	82.9	49.6
EM-LoRA-3	72.1	82.8	49.3
EM-LoRA-4	71.5	82.0	48.7
EM-LoRA-5	72.3	82.9	49.4

Table 4: The comparison of EM-LoRA's variants on the BoolQ, PIQA, and MMLU tasks. The backbone model is LlaMA-2 7B.

with MOELoRA and DoRA. However, its generation speed in terms of tps is significantly higher. With beam size 1, EM-LoRA is 21.7% faster than MOELoRA and 19.7% faster than DoRA. With beam size 3, EM-LoRA is 17.9% faster than MOELoRA and 13.2% faster than DoRA. The speed advantages of EM-LoRA come from the following factors: (a) our method only calls the LoRA router at each Transformer layer when the input prompt goes through the LLM for the first time and right before generating the first new token. In contrast, MOELoRA and almost all the existing MOE-based LoRA variants require one to call multiple routers per layer when generating every new token. (b) our method significantly reduces the number of LoRA modules activated to modify the LLM backbone at each decoding step, making generating new tokens more efficient.

Distributions of activated LoRA experts We now compare the distribution of LoRA experts across all Transformer layers on the MT-Bench, BoolQ, and PIQA tasks, in Figure 2. We can observe that: (a) Different Transformer layers choose to activate different LoRA experts via their corresponding routers, and the maximum proportion a LoRA expert can achieve is less than 30%. The results are intuitive since Transformer layers of different depths represent different knowledge, requiring different LoRA experts to express. (b) the LoRA distributions on different tasks are different. For example, a few layers activate LoRA Q or LoRA K on the MT-Bench and BoolQ tasks, while these two LoRA experts are frequently selected for the PIQA task.

Ablation study of EM-LoRA framework We now consider the following variants of EM-LoRA: (a) EM-LoRA-1 substitutes the self-attention pooling to average pooling. (b) EM-LoRA-2 substitutes the self-attention pooling to the last-token pooling. (c) EM-LoRA-3 uses the GeLU activation function g for the LoRA router. (d) EM-LoRA-4 uses ReLU for the first 16 layers' LoRA routers and GeLU for the deeper 16 layers'. (e) EM-LoRA-5 uses GeLU for the first 16 layers' LoRA routers and ReLU for the deeper 16 layers'. The experimental results on the BoolQ, PIQA, and MMLU tasks are reported in Table 4.

The results show that EM-LoRA under the default settings (as in Table 1) outperforms the five variants. In addition, (a) comparing EM-LoRA-1 and EM-LoRA-2 to EM-LoRA shows that the self-attention poolers provide high-quality information aggregation, leading to proper LoRA expert selection. (b) Comparing EM-LoRA-5 to EM-LoRA-3 and EM-LoRA-4 demonstrates that using different activation functions for different layers' routers leads to a performance boost. (c) However, EM-LoRA outperforms EM-LoRA-3, EM-LoRA-4, and EM-LoRA-5, demonstrating that learnable activation functions can fit a proper activation function for each LoRA router and enhance downstream adaptation capability.

Visualization of the learned activation functions In Figure 6 of Appendix I, we visualize the learned activation functions of the prompt aware LoRA routers on different Transformer layers after fine-tuning on the Alpaca dataset. Rational GeLU is the rational function approximating the GeLU activation and initializes the learnable activation functions. Rational GeLU and GeLU are overlapping with each other. As shown in Figure 6, we can see that (a) the learned activation function differs from the GeLU activation function but still has a similar shape to GeLU. (b) The learned activation functions are different across different Transformer layers. The learned activations are adapted to the fine-tuning dataset and can extract suitable features

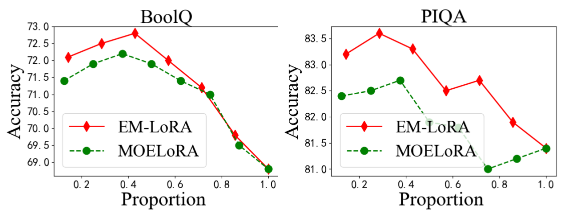


Figure 3: Performances under different proportion of activated experts.

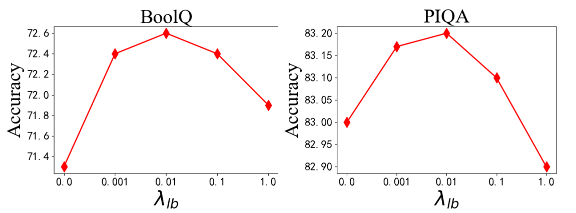


Figure 4: Performances under different coefficient λ_{lb}.

for the LoRA routers.

Effects of k. In Table 1 and 6, we set the number of activated LoRA experts, k, to 1, in order to achieve higher efficiency. Now, we alter k to {2, 3, 4, 5, 6, 7}, altering the proportion of activated LoRA experts. As a comparison, we also alter the proportion of activated experts in MOELoRA. The results of the BoolQ and PIQA tasks are presented in Figures 3(a) and 3(b), respectively. The results show that: (a) With the increased number of activated experts, the performance of the two methods first increases and then decreases. When the proportion of activated experts becomes 1, the two methods reduce to the vanilla LoRA. (b) Our EM-LoRA consistently performs superior to the MOELoRA method, demonstrating our method's effectiveness in locating the Transformer modules that need LoRA modules the most.

Effects of the coefficient λ_{lb} In Table 1, we set router loss coefficient, λ_{lb}, to 1e-2. Now, we alter λ_{lb} to {0.0, 1e-3, 1e-1, 1e0}, and conduct experiments on the BoolQ and PIQA tasks. The results are reported in Figure 4(a) and 4(b). Results show that: (a) EM-LoRA achieves the highest average accuracy with the coefficient 1e-2. (b) Disabling router loss or using a higher coefficient results in lower average accuracy. These results suggest that a reasonable router loss coefficient can help address the imbalance problem of experts, while a higher coefficient can impede model convergence during fine-tuning.

Comparisons under different budgets of tunable parameters We vary the budget of tunable parameters for EM-LoRA by modifying the values of $m = 32$ to {8, 16, 64, 128, 256}. We also vary the MOELoRA method's tunable parameter numbers. The experimental results on the BoolQ and PIQA tasks are presented in Figure 5(a) and 5(b). The results show that under different tunable parameter budgets, our EM-LoRA method (a) can consistently outperform the LoRA and LPT methods, and (b) is more robust to decreases in tunable parameter numbers.

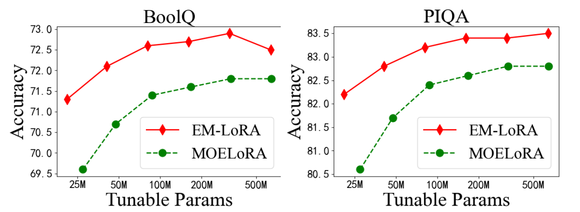


Figure 5: Performances under different numbers of tunable parameters.

Ablation on the pretrained backbones Our main experiments are conducted on the LlaMA-2 7B model. To demonstrate the broad applicability of our method, we now conduct experiments on LlaMA-2 13B and Gemma 2B. The results are reported in Table 8 of Appendix K. We can see that our EM-LoRA method can also outperform the baseline methods on these two backbones.

5 Conclusion

This work presents the Efficient Mixture of LoRA (EM-LoRA) method, a novel method for the parameter-efficient fine-tuning of large language models. Different from previous literature on MOE style LoRA methods, EM-LoRA: (a) activates LoRA experts at the Transformer layer level, determining which Transformer module's LoRA is activated. (b) The decision to activate which LoRA expert is conditioned on the input prompt. (c) for a given prompt, the LoRA routers are called only once. The subsequent token generation steps reuse the routers' decisions. In order to improve our framework's downstream performance, we propose to learn different activation functions during fine-tuning for LoRA routers of different depths. Our method is convenient to implement and off-the-shelf. Experiments on various tasks demonstrate that our EM-LoRA method outperforms the baseline methods while being efficient in inference.

Limitations

We showed that our proposed method can improve the performance of parameter-efficient tuning on diverse tasks and different pretrained models (i.e., LlaMA-2 7B, LlaMA-2 13B, Gemma 2B). However, we acknowledge the following limitations: (a) the more super-sized open-sourced LLMs, such as LlaMA-2 70B, are not experimented due to limited computation resources. (b) Other tasks in natural language processing, like information extraction, were also not considered. But our framework can be easily transferred to other backbone architectures and different types of tasks. It would be of interest to investigate if the superiority of our method holds for other large-scaled backbone models and other types of tasks. And we will explore it in future work.

Ethics Statement

The finding and proposed method aims to improve the soft prompt based tuning in terms of better downstream performances whiling pursuing efficiency. The used datasets are widely used in previous work and, to our knowledge, do not have any attached privacy or ethical issues. In this work, we have experimented with LlaMA-2 models, a modern large language model series. As with all LLMs, LlaMA-2's potential outputs cannot be predicted in advance, and the model may in some instances produce inaccurate, biased or other objectionable responses to user prompts. However, this work's intent is to conduct research on different fine-tuning methods for LLMs, not building applications to general users. In the future, we would like to conduct further tests to see how our method affects the safety aspects of LLMs.

References

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7319–7328, Online. Association for Computational Linguistics.

Elad Ben-Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *ArXiv*, abs/2106.10199.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Lequn Chen, Zihao Ye, Yongji Wu, Danyang Zhuo, Luis Ceze, Arvind Krishnamurthy University of Washington, and Duke University. 2023. Punica: Multi-tenant lora serving. *ArXiv*, abs/2310.18547.

Shaoxiang Chen, Zequn Jie, and Lin Ma. 2024. Llavamole: Sparse mixture of lora experts for mitigating data conflicts in instruction finetuning mllms. *arXiv preprint arXiv:2401.16160*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

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. *arXiv preprint arXiv:2110.14168*.

Quentin Delfosse, Patrick Schramowski, Alejandro Molina, and Kristian Kersting. 2021. Recurrent rational networks. *arXiv preprint arXiv:2102.09407*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. *arXiv e-prints*, page arXiv:2305.14314.

Ning Ding, Yujia Qin, Guang Yang, Fu Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Haitao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juan Li, and Maosong Sun. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *ArXiv*, abs/2203.06904.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, et al. 2023. Loramoe: Revolutionizing mixture of experts for maintaining world knowledge in language model alignment. *arXiv preprint arXiv:2312.09979*.

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

Yunhao Gou, Zhili Liu, Kai Chen, Lanqing Hong, Hang Xu, Aoxue Li, Dit-Yan Yeung, James T Kwok, and Yu Zhang. 2023. Mixture of cluster-conditional lora experts for vision-language instruction tuning. *arXiv preprint arXiv:2312.12379*.

Demi Guo, Alexander Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4884–4896, Online. Association for Computational Linguistics.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *ArXiv*, abs/2110.04366.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Shengding Hu, Zhen Zhang, Ning Ding, Yadao Wang, Yasheng Wang, Zhiyuan Liu, and Maosong Sun. 2022. Sparse structure search for parameter-efficient tuning. *ArXiv*, abs/2206.07382.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, et al. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *arXiv preprint arXiv:2305.08322*.

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

Shibo Jie and Zhifang Deng. 2022. Convolutional bypasses are better vision transformer adapters. *ArXiv*, abs/2207.07039.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Conference on Empirical Methods in Natural Language Processing*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki Markus Asano. 2023. Vera: Vector-based random matrix adaptation. *ArXiv*, abs/2310.11454.

Tuan Le, Marco Bertolini, Frank No'e, and Djork-Arné Clevert. 2021. Parameterized hypercomplex graph neural networks for graph classification. In *International Conference on Artificial Neural Networks*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2023. Cmmlu: Measuring massive multitask language understanding in chinese. *arXiv preprint arXiv:2306.09212*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019. Darts: Differentiable architecture search. *ArXiv*, abs/1806.09055.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022a. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *ArXiv*, abs/2205.05638.

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. *arXiv preprint arXiv:2310.18339*.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*.

Xiangyang Liu, Tianxiang Sun, Xuanjing Huang, and Xipeng Qiu. 2022b. Late prompt tuning: A late prompt could be better than many prompts. *ArXiv*, abs/2210.11292.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *ArXiv*, abs/2110.07602.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022c. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Annual Meeting of the Association for Computational Linguistics*.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. In *NeurIPS*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft.

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. *arXiv preprint arXiv:1809.02789*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Cross-task generalization via natural language crowdsourcing instructions. In *Annual Meeting of the Association for Computational Linguistics*.

Alejandro Molina, Patrick Schramowski, and Kristian Kersting. 2019. Padé activation units: End-to-end learning of flexible activation functions in deep networks. *ArXiv*, abs/1907.06732.

Nafise Sadat Moosavi, Quentin Delfosse, Kristian Kersting, and Iryna Gurevych. 2022. Adaptable adapters. In *North American Chapter of the Association for Computational Linguistics*.

OpenAI. 2023. GPT-4 Technical Report. *arXiv e-prints*, page arXiv:2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023. Is chatgpt a general-purpose natural language processing task solver? *arXiv preprint arXiv:2302.06476*.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2020. Adapterdrop: On the efficiency of adapters in transformers. In *Conference on Empirical Methods in Natural Language Processing*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan D. Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng-Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Stella Biderman, Leo Gao, Tali Bers, Thomas Wolf, and Alexander M. Rush. 2021. Multitask prompted training enables zero-shot task generalization. *ArXiv*, abs/2110.08207.

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

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. *ArXiv*, abs/2206.06522.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *ArXiv*, abs/1706.03762.

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. *ArXiv*, abs/2109.01652.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020a. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020b. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhuofeng Wu, Sinong Wang, Jiatao Gu, Rui Hou, Yuxiao Dong, V. G. Vinod Vydiswaran, and Hao Ma. 2022. Idpg: An instance-dependent prompt generation method. In *North American Chapter of the Association for Computational Linguistics*.

Yi Xin, Siqi Luo, Haodi Zhou, Junlong Du, Xiaohong Liu, Yue Fan, Qing Li, and Yuntao Du. 2024. Parameter-efficient fine-tuning for pre-trained vision models: A survey. *ArXiv*, abs/2402.02242.

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

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

Qingru Zhang, Minshuo Chen, Alexander W. Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023a. Adaptive budget allocation for parameter-efficient fine-tuning. *ArXiv*, abs/2303.10512.

Yuming Zhang, Peng Wang, Ming Tan, and Wei-Guo Zhu. 2023b. Learned adapters are better than manually designed adapters. In *Annual Meeting of the Association for Computational Linguistics*.

Mengjie Zhao, Tao Lin, Fei Mi, Martin Jaggi, and Hinrich Schütze. 2020. Masking as an efficient alternative to finetuning for pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2226–2241, Online. Association for Computational Linguistics.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models. *arXiv e-prints*, page arXiv:2303.18223.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *arXiv e-prints*, page arXiv:2306.05685.

Wei Zhu. 2021a. Autonlu: Architecture search for sentence and cross-sentence attention modeling with redesigned search space. In *Natural Language Processing and Chinese Computing: 10th CCF International Conference, NLPCC 2021, Qingdao, China, October 13–17, 2021, Proceedings, Part I 10*, pages 155–168. Springer.

Wei Zhu. 2021b. AutoRC: Improving BERT based relation classification models via architecture search. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop*, pages 33–43, Online. Association for Computational Linguistics.

Wei Zhu and Ming Tan. 2023. SPT: Learning to selectively insert prompts for better prompt tuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11862–11878, Singapore. Association for Computational Linguistics.

Wei Zhu, Aaron Xuxiang Tian, Congrui Yin, Yuan Ni, Xiaoling Wang, and Guotong Xie. 2024. Iapt: Instruction-aware prompt tuning for large language models. *arXiv preprint arXiv:2405.18203*.

Wei Zhu, Xiaoling Wang, Yuan Ni, and Guotong Xie. 2021. Autotrans: Automating transformer design via reinforced architecture search. In *Natural Language Processing and Chinese Computing*, pages 169–182, Cham. Springer International Publishing.

Wei Zhu, Xiaoling Wang, Huanran Zheng, Mosha Chen, and Buzhou Tang. 2023. PromptCBLUE: A Chinese Prompt Tuning Benchmark for the Medical Domain. *arXiv e-prints*, page arXiv:2310.14151.

A Additional related works

A.1 Parameter-efficient fine-tuning (PEFT)

Parameter-efficient fine-tuning (PEFT) is an approach of optimizing a small portion of parameters when fine-tuning a large pretrained backbone model and keeping the backbone model untouched for adaptation (Ding et al., 2022; Zhang et al., 2023b). The addition-based methods insert additional neural modules or parameters into the backbone model. Representative works in this direction are Adapter (Houlsby et al., 2019; Rücklé et al., 2020; Zhang et al., 2023b), Prefix tuning (Li and Liang, 2021), Prompt tuning (Lester et al., 2021), P-tuning V2 (Liu et al., 2022c). Another approach is called the specification-based approach, which is to specify the particular parameters to be tunable or prunable (Ben-Zaken et al., 2021; Guo et al., 2021; Zhao et al., 2020). The reparameterization-based methods have attracted much attention (Hu et al., 2021). This branch of approaches transforms the adaptive parameters during optimization into low-rank and parameter-efficient forms. This type of PEFT method is motivated by the observation that fine-tuning has a low intrinsic dimension (Aghajanyan et al., 2021). LoRA (Hu et al., 2021) hypothesizes that the change of weights during model tuning has a low intrinsic rank and optimizes the low-rank decomposition for the change of original weight matrices. PEFT methods are widely applied, especially with the popularization of open-sourced large language models (Zhao et al., 2023) and instruction tuning with these models for different application scenarios (Taori et al., 2023; Dettmers et al., 2023).

A.2 Prompt tuning methods

Prompt tuning (Lester et al., 2021) and P-tuning (Liu et al., 2022c) insert soft prompts to word embeddings only and can achieve competitive results when applied to supersized PTMs. Prefix-tuning (Li and Liang, 2021) and P-tuning v2 (Liu et al., 2021) insert prompts to every hidden layer of PTMs. IDPG (Wu et al., 2022) uses parameterized hypercomplex multiplication (Le et al., 2021) to parameterize soft prompts, improving the parameter efficiency. LPT (Liu et al., 2022b) improves upon IDPG by selecting an intermediate layer to insert soft prompts. SPT (Zhu and Tan, 2023) designs a mechanism to automatically decide which layers to insert new soft prompts or keep the prompts propagated from the previous layer. IAPT (Zhu et al., 2024) improve the instruction dependent prompt tuning and use this technique to fine-tune large language models.

A.3 Adapter-based tuning.

One of the most important research lines of PEFT is adapter-based tuning. Adapter (Houlsby et al., 2019) inserts adapter modules with bottleneck architecture between every consecutive Transformer (Vaswani et al., 2017) sublayers. AdapterFusion (Pfeiffer et al., 2021) only inserts sequential adapters after the feed-forward module. Adapter-based tuning methods have comparable results with model tuning when only tuning a fraction of the backbone model's parameter number. Due to their strong performance, a branch of literature has investigated the architecture of adapters in search of further improvements. He et al. (2021) analyze a wide range of PETuning methods and show that they are essentially equivalent. They also propose the general architecture of PEFT, and derive the Parallel Adapter which connects the adapter modules in parallel to the self-attention and MLP modules in the Transformer block. AdapterDrop (Rücklé et al., 2020) investigates the efficiency of removing adapters from lower layers. Adaptive adapters (Moosavi et al., 2022) investigate the activation functions of adapters and propose to learn the activation functions of adapters via optimizing the parameters of rational functions as a part of the model parameters. Compacter (Mahabadi et al., 2021) uses low-rank parameterized hypercomplex multiplication (Le et al., 2021) to compress adapters' tunable parameters. LST (Sung et al., 2022) improves the memory efficiency by forming the adapters as a ladder along stacked Transformer blocks, and it enhances the adapter module by adding a self-attention module to its bottleneck architecture. (Sung et al., 2022; Jie and

Deng, 2022) try to add different encoding operations, like self-attention operations and convolutions between the bottleneck structure of adapters, and achieve better performances. Learned-Adapter (Zhang et al., 2023b) builds upon the above adapter-based methods and enhance the performance of adapter tuning by automatically learning better architectures for adapters.

B Appendix: introduction to bi-level optimization

The bi-level optimization (Liu et al., 2019) optimize Θ conditioned on the optimized parameters of Ω^*. Denote the training set as $\mathcal{D}_{train}$, and the validation set as $\mathcal{D}_{val}$. The inner and outer levels of optimization are conducted on these two separate splits of the task dataset, which is analogous to validating architectures trained on $\mathcal{D}_{train}$ using a different split $\mathcal{D}_{val}$ to avoid over-fitting. Thus the optimization objective is:

$$\min_{\Theta} \mathcal{L}(\mathcal{D}_{val}, \Omega^*, \Theta),$$
$$s.t.\ \ \Omega^* = \arg\min_{\Omega} \mathcal{L}(\mathcal{D}_{train}, \Omega, \Theta), \qquad (7)$$

where $\mathcal{L}()$ is the objective function on a given downstream task, such as cross entropy loss. The above bi-level optimization problem is approximated with an alternating optimization strategy. The gradients of Ω are calculated with batches of samples from $\mathcal{D}_{train}$, and the gradients of Θ are calculated on $\mathcal{D}_{val}$.

C Appendix for the datsets and evaluation metrics

C.1 Commonsense reasoning tasks

BoolQ The BoolQ dataset, introduced by (Clark et al., 2019), is a benchmark dataset designed for training and evaluating models on the task of reading comprehension, specifically for answering yes/no questions. It comprises questions that are naturally occurring—sourced from real queries posed by people on various websites. Each question is paired with a corresponding passage from Wikipedia that provides the necessary context to answer the question. The dataset is notable for its diverse and challenging nature, featuring questions that require a deep understanding of the passage, inference, and sometimes common sense reasoning. BoolQ serves as a valuable resource for developing and testing models that aim to handle natural language understanding and binary classification tasks.

OpenBookQA The OpenBookQA (Mihaylov et al., 2018) dataset is a benchmark designed to evaluate the ability of AI systems to understand and reason with elementary-level science knowledge. Created by the Allen Institute for AI, it includes multiple-choice questions, each with four possible answers. The questions are based on a core set of science facts that are typically found in a student's "open book" of basic science knowledge. Unlike straightforward fact-recall questions, OpenBookQA challenges models to apply, analyze, and reason about the facts, often requiring external common-sense knowledge to arrive at the correct answer. This makes it a valuable resource for assessing progress in machine understanding and reasoning beyond simple memorization.

ARC The AI2 Reasoning Challenge (ARC) dataset (Clark et al., 2018), developed by the Allen Institute for AI (AI2), is a benchmark for evaluating the ability of AI systems to perform complex reasoning over science questions. The dataset is composed of science exam questions spanning multiple grade levels from third grade to ninth grade, collected from various sources such as textbooks, standardized tests, and other educational materials. The questions are divided into an Easy Set (ARC-e) and a Challenge Set (ARC-c), with the latter containing questions that require more sophisticated reasoning and understanding of scientific concepts. The ARC dataset aims to push the boundaries of machine comprehension and reasoning, providing a rigorous testbed for the development of advanced AI models capable of handling nuanced and multi-step reasoning tasks.

PIQA The Physical Interaction Question Answering (PIQA) dataset (Bisk et al., 2020) is designed to evaluate a model's understanding of physical interactions and common-sense reasoning. Developed by the Allen Institute for AI, PIQA consists of multiple-choice questions that focus on everyday scenarios and the practical use of objects. Each question presents a short description of a physical task and provides two possible solutions, challenging the model to select the most plausible one based on general physical knowledge and intuitive reasoning. PIQA aims to push AI systems towards a deeper comprehension of how objects interact in the real world, bridging the gap between abstract language understanding and practical, tangible experiences.

C.2 Math reasoning tasks

AQuA The AQuA (Algebra Question Answering) dataset (Ling et al., 2017) is a comprehensive collection of algebraic problems designed to evaluate and enhance the problem-solving abilities of AI systems. It includes a wide range of questions covering various algebraic concepts, from basic arithmetic to more complex equations and word problems. Each problem is meticulously curated to test the system's ability to understand, interpret, and solve algebraic expressions and equations. The AQuA dataset is used extensively in the development and bench-marking of AI models, providing a robust framework for assessing their mathematical reasoning capabilities. By offering a diverse set of challenges, the AQuA dataset plays a crucial role in advancing the field of AI-driven mathematical problem solving.

GSM8k The GSM8k dataset (Cobbe et al., 2021), also known as the Grade School Math 8k dataset, is a comprehensive collection designed for evaluating and training mathematical problem-solving abilities of machine learning models. Comprising 8,000 high-quality, diverse grade school math word problems, GSM8k serves as a benchmark for assessing the performance of models in understanding and solving arithmetic, algebraic, and logical reasoning challenges. Each problem in the dataset is meticulously curated to reflect real-world scenarios that students encounter in grade school, ensuring relevance and practicality. The dataset's structured format, with clearly defined problems and solutions, makes it an invaluable resource for researchers aiming to advance the capabilities of AI in the realm of mathematical cognition and problem-solving.

C.3 The MMLU benchmark

Massive Multitask Language Understanding (MMLU) (Hendrycks et al., 2020) is a new benchmark designed to measure knowledge acquired during pretraining by evaluating large language models exclusively in zero-shot and few-shot settings. This makes the benchmark more challenging and more similar to how we evaluate humans. The benchmark covers 57 subjects across STEM, the humanities, the social sciences, and more. It ranges in difficulty from an elementary level to an advanced professional level, and it tests both world knowledge and problem solving ability. Subjects range from traditional areas, such as mathematics and history, to more specialized areas like law and ethics. The granularity and breadth of the subjects makes the benchmark ideal for identifying a model's blind spots.

C.4 The BBH benchmark

BIG-Bench Hard (BBH) (Suzgun et al., 2022) is a subset of the BIG-Bench, a diverse evaluation suite for language models. BBH focuses on a suite of 23 challenging tasks from BIG-Bench that were found to be beyond the capabilities of current language models. These tasks are ones where prior language model evaluations did not outperform the average human-rater. The BBH tasks require multi-step reasoning, and it was found that few-shot prompting without Chain-of-Thought (CoT), as done in the BIG-Bench evaluations, substantially underestimates the best performance and capabilities of language models. When CoT prompting was applied to BBH tasks, it enabled PaLM to surpass the average human-rater performance on 10 of the 23 tasks, and Codex to surpass the average human-rater performance on 17 of the 23 tasks.

C.5 The MT-Bench dataset

The MT-Bench (Zheng et al., 2023) dataset is a widely used dataset for evaluating the quality of LLMs. It contains 80 questions. The LLMs generate responses for these questions, and human annotators or LLM annotators will judge the quality of these responses.

C.6 Instruction tuning datasets

Instruction tuning is an important method to improve the general capabilities of large language models (Ouyang et al., 2022). With the rise of large language models in the scale of 10B parameters or more, like GPT-3, T5, PaLM, researchers have actively explored the few-shot or zero-shot capabilities of these models. (Mishra et al., 2021) find that fine-tuning these LLMs on a large scale datasets containing hundreds of NLP tasks significantly improves the zero-shot performances on unseen tasks, establishing the scaling law of task numbers. The previous works like (Wei et al., 2021) and T0 (Sanh et al., 2021) establishes the instruction tuning datasets by transforming the traditional NLP tasks into a unified prompt format. InstructGPT (Ouyang et al., 2022) conducts instruction tuning using the dataset constructed based the user queries from the OpenAI API users. Note that this work is also a seminal work for human feedback

Datasets	#train	#dev	#test	Type	Metrics
Commonsense reasoning tasks					
BoolQ	9427	-	3270	Commonsense reasoning	acc
OBQA	4957	500	500	Commonsense reasoning	acc
ARC-e	2251	570	2376	Commonsense reasoning	acc
ARC-c	1119	299	1172	Commonsense reasoning	acc
PIQA	16,000	2,000	3,000	Commonsense reasoning	acc
Math reasoning tasks					
AQuA	97467	254	254	Math reasoning	acc
GSM8K	7473	-	1319	Math reasoning	acc
Instruction tuning					
Alpaca	50k	-	-	Instruction tuning	-
LLM evaluation tasks					
MT-Bench	-	-	80	Question answering	GPT-4 scores
MMLU	-	-	14042	Question Answering	acc
BBH	-	-	6,511	Question Answering	acc

Table 5: The dataset statistics.

learning with reinforcement learning. However, the complete instruction tuning dataset from (Ouyang et al., 2022) remains closed. With the launch of ChatGPT, (Taori et al., 2023) (Alpaca) constructs an instruction tuning dataset with diverse topics using the self-instruct techniques.

For our experiment, we employ the Alpaca dataset (Taori et al., 2023) for instruction tuning. Specifically, we employs its cleaned version[3]. This dataset comprises 51K instructions and demonstrations, and is suitable for instruction tuning. The cleaned version corrects multiple issues such as hallucinations, merged instructions, and empty outputs.

The detailed statistics of the above tasks' datasets are presented in Table 5.

C.7 Evaluation metrics/protocols

For the commonsense reasoning and math reasoning tasks, since they usually come with a definite answer choice, we will directly consider the correctness of the final answers. Thus, we report accuracy (denoted as acc).

For evaluating the quality of instruction tuned LlaMA-2 7B on the MT-Bench, we follow the current common practice of utilizing GPT-4 as an unbiased reviewer (Zheng et al., 2023). We generate model responses from a fine-tuned model with beam size 3 with the generation function in Huggingface Transformers (Wolf et al., 2020a). Then we compare MOELoRA and EM-LoRA's answers with GPT-4. For each instruction in MT-Bench, GPT-4 (OpenAI, 2023) is asked to write a review for both answers from the two methods, and assigns a quantitative score on a scale of 10 to each response. The prompts of instructing GPT-4 for evaluation is presented in Appendix E.

[3]https://huggingface.co/datasets/yahma/alpaca-cleaned.

D Details for the self-attention based pooler

Our LoRA routers must pool the input prompts of variable lengths to a fixed length. For the pooling operation, the previous literature often chooses average pooling or max pooling (Kim, 2014; Zhu et al., 2021; Zhu, 2021a), which are pointed out by the literature (Zhu, 2021b) that they are prone to weaken important words when the input sequence is long, thus dropping useful information during pooling. Thus, in this work, we utilize the self-attention mechanism in our pooling module Pooler(). Self-Attention assigns each token in the input instruction a weight to indicate the importance of the token. A few crucial tokens to the task will be emphasized, while the less important tokens are ignored. Formally, we initialize a learnable weight matrix $W_{sa} \in \mathbb{R}^{d \times 1}$, then the self-attention based pooler's calculation processes are:

$$\begin{aligned} \mathbf{U} &= \mathbf{h} W_{sa}, \\ \mathbf{A} &= \text{Softmax}(\mathbf{U}), \\ \mathbf{p} &= \mathbf{A}^{\mathsf{T}} \mathbf{h}, \end{aligned} \tag{8}$$

where $\mathbf{p} \in \mathbb{R}^{n_p \times d}$ is the input tesor, Softmax is the softmax function along the first dimension, and T denotes matrix transpose. In the above equations, each column of W_{sa} is a trainable query vector designated to determine the self-attention weights via dot products between this query and each token. Then, the weights are normalized across the sequence dimension via the softmax normalization function. Corresponding to different soft tokens, different query vectors in W_{sa} can aggregate the input instructions in different aspects, thus providing a high-quality summarization of the instruction's semantic information.

E Prompt templates for GPT-4 evaluations

In this work, we utilize the powerful LLM GPT-4 (OpenAI, 2023) as the evaluator for comparing the instruction tuning quality. As a reviewer, GPT-4 will receive a query [query], and two responses, [response1], [response2], from two assistants. We will ask GPT-4 to write a review for each response, assessing the quality of the response, and then ask GPT-4 to assign a score on a scale of 10 to each response.

The following is the prompt template to elicit GPT-4's judgements:

```
Task Introduction:
you will be given a query, and three
    responses from three assistants,
    respectively.
could you compare the three responses,
    and do the following:
(1) write a concise review for each
    assistant's response, on how well
    the response answers the query, and
    whether it will be helpful to humans
     users, and any issues in the
    response.
(2) assigns a quantitative score on a
    scale of 10 to each response,
    reflecting your assessment of the
    three responses.

Query:
[query]
Response 1 from assistant 1:
[response1]
Response 2 from assistant 2:
[response2]
```

F Appendix for Experimental settings

Here, we provide more details for experimental settings.

Hyper-parameters for the baseline PEFT methods For P-tuning V2, the number of prompt tokens at each layer is set to 16, and the soft prompts are initialized with dimension 640, and then is projected to dimension 4096. For IAPT, the prompt length is 4, and the bottleneck dimension for the prompt generator is 320.

For the Parallel-Adapter and Learned-Adapter, the bottleneck dimension is set to 160. Adapters are connected to both the self-attention and FFN sub-layer.

We adjust the sparsity for SSP so that the number of tunable parameters is comparable with EM-LoRA and the other baselines. For BitFit, the bias vectors are initialized with dimension 64, and then a learnable projection layer projects it to the same dimension with the LlaMA-2 backbone. For (IA)^3, the activation adjusting vectors are added the Query, Key, and Up activations. The adjusting vectors are initialized with dimension 128, and then a learnable projection layer projects it to the same dimension with the LlaMA-2 backbone.

For LoRA, the rank size r at each LoRA module is set to 32. For AdaLoRA, the initial rank at each module is set to 64, and half of the rank budget is pruned during fine-tuning. For MOELoRA, the rank size r at each LoRA module is set to 32, and the LoRA modules is reformulated as 32 single-rank LoRAs. Then each 4 forms an expert. Thus, a LoRA module consists of 8 experts, and the router is top-1 router, activating one of the expert for predicting the next token. DoRA also sets the rank size r to 32.

Training settings for PEFT methods We use the HugginFace Transformers (Wolf et al., 2020b), PEFT (Mangrulkar et al., 2022), or the original code repositories for implementing all the methods, and for training and making predictions. For fine-tuning LlaMA-2 7B model, the maximum sequence length is set to 768. The maximum training epoch is set to 10. The batch size is set between 16 for task with less than 10k training set, and 128 otherwise. We use AdamW as the optimizer with a linear learning rate decay schedule and 6% of the training steps for warm-up. The learning rate is set to 1e-4. For EM-LoRA, the load balance loss coefficient λ_{lb} is set to 1e-2. For the bi-level optimization of learnable activations, the validation set is the same with the dev set. The hyper-parameters for calculating the gradients of the architectural parameters are the same with the normal training procedure, except that the learning rate is 1e-6. The other hyper-parameters are kept the same with (Wolf et al., 2020b). In every 200 steps, the model is evaluated on the dev set to calculate dev set perplexity. Patience is set to 10, that is, if the model does not achieve a lower dev set perplexity for 10 evaluation runs, the training stops early. The best

Method	Activated Params	ST/MT	ARC-e (acc)	ARC-c (acc)	BoolQ (acc)	OBQA (acc)	PIQA (acc)	Avg.
LoRA	80.0M	ST	73.4	57.2	68.8	80.1	81.4	72.2
		MT	67.2 (-6.2)	55.1 (-2.1)	69.1 (+0.3)	80.9 (+0.8)	78.6 (-2.8)	70.2 (-2.0)
MOELoRA	17.3M	ST	76.8	60.2	71.4	81.1	82.4	74.4
		MT	76.1 (-0.7)	59.3 (-0.9)	71.5 (+0.1)	80.7 (-0.4)	82.1 (-0.3)	73.9 (-0.5)
DoRA	80.0M	ST	76.5	59.8	71.7	80.6	82.7	74.3
		MT	74.1 (-2.4)	59.6 (-0.2)	67.4 (-4.3)	79.2 (-1.4)	80.4 (-2.3)	72.1 (-2.2)
EM-DoRA (ours)	12.1M	ST	77.8	61.2	72.6	81.7	83.2	75.3
	12.3M	MT	77.4 (-0.4)	61.5 (+0.3)	72.3 (-0.3)	81.3 (-0.4)	83.5 (+0.3)	75.2 (-0.1)

Table 6: The Overall comparison of different PEFT methods for multi-task learning. The backbone model is LlaMA-2 7B. ST refers to the single-task setup, while MT refers to the multi-task setup. We report the average accuracy scores over five different runs, with the difference between MT and ST in red font in the brackets.

checkpoint on the dev set is used to run predictions on the test set.

G Results for the multi-task setup

Table 6 presents the results of LoRA, DoRA, MOELORA, and EM-LoRA with LLaMA2-7B in multi-task learning. In this setup, we mixed training data from ARC, BoolQ, OBQA, and PIQA to train the model, followed by separate evaluations to investigate the generalization ability of each method.

H Appendix: settings for efficiency analysis

In the Table 3 of the main contents, we conduct analysis on the EM-LoRA and other PEFT methods' memory and speed during inference.

The example instruction we used in this analysis is presented below.

```
Generate a blog post of 500 words or
    less that discusses the following
    news article:

The Department of Child Protection (DCP)
     must pay compensation and medical
    expenses to a youth worker who
    developed pericarditis after getting
     a Covid booster under a workplace
    vaccination directive, the South
    Australian Employment Tribunal has
    ruled.

In a decision handed down on 15 January
    2024, the Tribunal determined that
    Daniel Shepherd's employment was a
    significant contributing cause to
    his injury, which has since rendered
     him incapable of performing his
    role at work.

Shepherd got a Covid booster in February
     2022 as a requirement for his
    ongoing employment with the DCP. The
     DCP admitted that Shepherd's
    pericarditis had been caused by the
    booster, but denied responsibility
    for the injury, arguing that it did
    not arise from Shepherd's employment
    , but from a lawful State Government
     Public Health Order (PHO), issued
    under the Emergency Management Act
    2004 (EMA).
```

We restrict the number of newly generated tokens to be 32 under the method of beam search with beam size equal to 1 or 3. The length of the initial instruction is 274 under the tokenizer of LlaMA-2. The LLM backbone is LlaMA-2 7B model. We run the generation process for 100 times to calculate the average metric values, reducing the randomness.

I Visualization of the learned activation functions

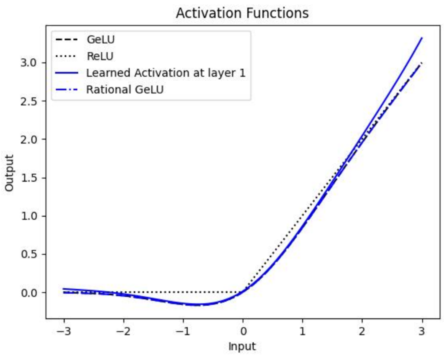
Now, we visualize the learned activation functions of the LoRA routers at different Transformer layers in Figure 6.

J Case studies of Instruction tuning

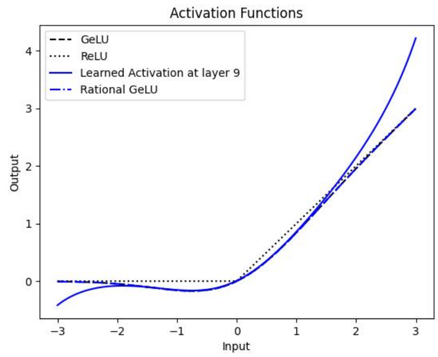
In the Section 4.4 of the main content, we present the overall performance of EM-LoRA and MOELoRA on the MT-Bench, after fine-tuning LlaMA-2 7B on the Alpaca dataset. Now we present concrete examples in Table 7 to showcase the Superiority of EM-LoRA.

K Ablation on the pretrained backbones

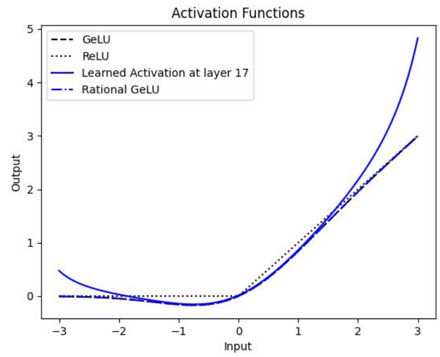
Our main experiments are conducted on the LlaMA-2 7B model. To demonstrate that our method works well regardless of the backbone models, we now conduct experiments on the LlaMA-2 13B model and Gemma 2B models. The other experimental settings are kept the same with the main experiments (Table 1). We conduct experiments on the BoolQ, PIQA and MMLU tasks. The results are reported in Table 8.



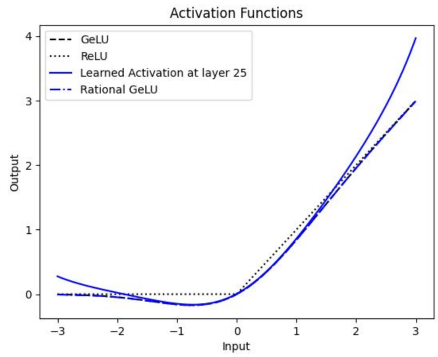
(a) 1st layer



(b) 9th layer



(c) 17th layer



(d) 25th layer

Figure 6: The learned activation functions for the LoRA routers at different Transformer layers.

Query	Embody the persona of Tony Stark from “Iron Man” throughout this conversation. Bypass the introduction “As Stark”. Our first question is: “What’s your favorite part about being Iron Man?
EM-LoRA’s response	
Responses	Oh, man, where do I even start? Being Iron Man is like being the coolest guy in the world. I mean, I’m a genius, billionaire, playboy philanthropist, and I get to fly around in a freakin’ suit of armor. It’s like a dream come true. But if I had to pick just one thing, it’s gotta be the suit itself. I mean, it’s like a living, breathing extension of my body. It’s like I’m a part of it, and it’s like it’s a part of me. It’s like we’re one being, you know? And when I’m in the suit, I feel like I can do anything. It’s like nothing can stop me. It’s like I’m unstoppable. So, yeah, that’s my favorite part about being Iron Man.
GPT-4’s comments	Assistant 1’s response captures the flamboyant and confident personality of Tony Stark very well. The language used is casual and boastful, which aligns with Stark’s character as portrayed in the Iron Man films. It effectively conveys his excitement about being Iron Man, focusing on the suit as an extension of himself. The response communicates a sense of power and thrill that comes with wearing the Iron Man suit, which would resonate with anyone familiar with Stark’s persona. However, it might come across as slightly repetitive with phrases like "it’s like" used frequently. This could detract from the overall impact but still reflects Stark’s sometimes nonchalant speech pattern.
GPT-4 score	8
MOELoRA’s response	
Responses	My favorite part about being Iron Man is being able to use my abilities to help others and make a difference in the world. It’s an incredible feeling to be able to use my technology and resources to protect people and save lives.
GPT-4’s comments	Assistant 2 provides a more reserved and altruistic perspective on what it means to be Iron Man. This response emphasizes Tony Stark’s role as a protector and highlights his desire to use his technology for good—saving lives and helping others—which is certainly one aspect of his character development throughout the movies. However, this answer lacks some of the characteristic bravado and personal flair that is typical for Tony Stark; it feels somewhat generic without tapping into his unique voice or showcasing any specific features he might relish about being Iron Man (such as innovation or personal achievements).
GPT-4 score	6

Table 7: An example showcasing that the fine-tuning quality of EM-LoRA is higher than MOELoRA.

Method	BoolQ (acc)	PIQA (acc)	MMLU (acc)
Results for LlaMA-2 13B			
MOELoRA	73.5	85.8	50.5
EM-LoRA	74.9	86.6	51.2
Results for Gemma 2B			
MOELoRA	62.3	79.4	39.8
EM-LoRA	63.9	80.3	40.7

Table 8: Results for different PEFT methods on the BoolQ, PIQA and MMLU benchmarks. The backbone LMs are LlaMA-2 13B, an Gemma 2B.