12-2023

LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models

Zhiqiang HU

Lei WANG

Yihuai LAN

Wanyu XU

Ee-peng LIM
Singapore Management University, eplim@smu.edu.sg

See next page for additional authors

Citation

Author

Zhiqiang HU, Lei WANG, Yihuai LAN, Wanyu XU, Ee-peng LIM, Lidong BING, Xing XU, Soujanya PORIA, and Roy Ka-Wei LEE

LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models

Zhiqiang Hu[1] **Lei Wang**[2] [*] **Yihuai Lan** **Wanyu Xu**[4] **Ee-Peng Lim**[2]
Lidong Bing[3] **Xing Xu**[5] **Soujanya Poria**[1] **Roy Ka-Wei Lee**[1]
[1]Singapore University of Technology and Design
[2]Singapore Management University
[3]DAMO Academy, Alibaba Group
[4]Southwest Jiaotong University
[5]University of Electronic Science and Technology of China

Abstract

The success of large language models (LLMs), like GPT-4 and ChatGPT, has led to the development of numerous cost-effective and accessible alternatives that are created by finetuning open-access LLMs with task-specific data (e.g., ChatDoctor) or instruction data (e.g., Alpaca). Among the various fine-tuning methods, adapter-based parameter-efficient fine-tuning (PEFT) is undoubtedly one of the most attractive topics, as it only requires fine-tuning a few external parameters instead of the entire LLMs while achieving comparable or even better performance. To enable further research on PEFT methods of LLMs, this paper presents LLM-Adapters, an easy-to-use framework that integrates various adapters into LLMs and can execute these adapter-based PEFT methods of LLMs for different tasks. The framework includes state-of-the-art open-access LLMs such as LLaMA, BLOOM, and GPT-J, as well as widely used adapters such as Series adapters, Parallel adapter, Prompt-based learning and Reparametrization-based methods. Moreover, we conduct extensive empirical studies on the impact of adapter types, placement locations, and hyper-parameters to the best design for each adapter-based methods. We evaluate the effectiveness of the adapters on fourteen datasets from two different reasoning tasks, Arithmetic Reasoning and Commonsense Reasoning. The results demonstrate that using adapter-based PEFT in smaller-scale LLMs (7B) with few extra trainable parameters yields comparable, and in some cases superior, performance to powerful LLMs (175B) in zero-shot inference on both reasoning tasks.

1 Introduction

Large language models (LLMs), such as ChatGPT (OpenAI, 2022) and GPT-4 (OpenAI, 2023), have demonstrated unprecedented performance across various natural language processing (NLP) tasks (Qin et al., 2023) and multi-modal tasks (Shen et al., 2023). These LLMs often possess sizes exceeding hundreds of billions of parameters and are closed-source. Consequently, this has spurred the development of accessible and cost-effective alternatives such as LLaMA (Touvron et al., 2023). These alternatives involve fine-tuning open-source LLMs utilizing either task-specific data (e.g., ChatDoctor (Yunxiang et al., 2023)) or instructional data (e.g., Alpaca (Taori et al., 2023)). However, full-model fine-tuning (FFT) is computationally and storage-intensive, thereby presenting significant challenges in practical implementation.

Prior to the emergence of FFT of LLMs (e.g., LLaMA), a compelling solution called parameter-efficient fine-tuning (PEFT) (Houlsby et al., 2019) has been proposed in the NLP field, specifically for pre-trained models (e.g., BERT (Devlin et al., 2018)), offering a promising approach for efficiently fine-tuning LLMs. The advantage of PEFT lies in its ability to fine-tune only a small set of external parameters rather than the entire backbone model while still achieving comparable or even superior performance (Mangrulkar et al., 2022). Moreover, PEFT can effectively mitigate catastrophic forgetting in comparison to FFT (Wang et al., 2022). As shown in Table 1, the advantage of PEFT has resulted in the developing of diverse PEFT modules, encompassing series adapters (Houlsby et al., 2019; Wang et al., 2022; He et al., 2022b; Fu et al., 2021), parallel adapters (He et al., 2022a), reparameterization-based methods (Hu et al., 2021; Edalati et al., 2022), and prompt-based learning methods (Lester et al., 2021; Li and Liang, 2021).

By incorporating these PEFT modules into backbone models (i.e., LLMs), we can capitalize on the remarkable capabilities of backbone models without requiring extensive computational resources. This opens up opportunities for a broader range of applications, enabling even those with limited ac-

[*]Corresponding author.

cess to high-performance computing to harness the power of LLMs in their specific tasks. Despite the success of PEFT for pre-trained models, it remains unclear which PEFT module, in combination with which layer and hyperparameter configuration, is most suitable for a given task or dataset when meeting LLMs (e.g., LLaMA (Touvron et al., 2023)). Therefore, further investigation is needed to determine the optimal PEFT setup that maximizes performance across different tasks and datasets.

Motivated by this, in this paper, we conduct a comprehensive empirical study of PEFT of three representative open-source LLMs, including BLOOM (Muennighoff et al., 2022), GPT-J (Wang and Komatsuzaki, 2021), and LLaMA (Touvron et al., 2023). Specifically, we undertake an empirical study to address the following three research questions: (i) What is the optimal placement and configuration of different PEFT methods? (ii) How's the performance of different adapters across downstream tasks? And (iii) What are the differences in performance between in-distribution (ID) and out-of-distribution (OOD) scenarios for PEFT methods? The findings of our study are as follows:

1. **The optimal placement for the series adapter, parallel adapter, and LoRA is after the MLP layers, parallel with the MLP layers, and located after both the Attention layers and MLP layers simultaneously, respectively**;
2. **Smaller language models with the PEFT approach can attain competitive or superior performance on specific tasks compared to larger language models. For instance, LLaMA-13B with LoRA can outperform GPT-3.5 (>175B) on MultiArith, AddSub, and SingleEq** ;
3. **The ID fine-tuned LLaMA-13B with adapters outperforms ChatGPT on commonsense reasoning tasks indicating that smaller language models have the potential to outperform larger language models on specific tasks with ID fine-tunig data.**

Our contributions can be summarized as follows:

- We conduct a comprehensive empirical study of various PEFT methods applied in different open-source LLMs.
- To facilitate our empirical study, we construct two high-quality training datasets to enhance PEFT performance in math reasoning and commonsense reasoning tasks.

Method	Prompt	Repara	Series	Parallel
Prompt Tuning (Lester et al., 2021)	√			
Prefix-Tuning (Li and Liang, 2021)	√			
Spot (Vu et al., 2021)	√			
IPT (Qin et al., 2021)	√			
LoRA (Hu et al., 2021)		√		
KronA (Edalati et al., 2022)		√		
Adapters (Houlsby et al., 2019)			√	
AdaMix (Wang et al., 2022)			√	
SparseAdapter (He et al., 2022b)			√	
LeTS (Fu et al., 2021)			√	
Parallel Adapter (He et al., 2022a)				√
MAM Adapter (He et al., 2021)	√	√	√	
UniPELT (Mao et al., 2021)	√	√	√	
Compacter (Henderson et al., 2021)		√	√	
S4-model (Chen et al., 2023)	√	√		

Table 1: The PEFT methods are categorized based on the four common basic methods. "Prompt" represents prompt-based learning methods, "Repara" denotes reparametrization-based methods, "Series" is Series Adapter, while "Parallel" represents Parallel Adapter.

- We develop a user-friendly framework, LLM-Adapter, seamlessly integrates diverse adapters into LLMs, empowering researchers to implement adapter-based PEFT methods for a wide range of tasks.
- We conduct extensive experiments to answer the three research questions to serve as inspiration for future research.

2 PEFT Overview

In this section, we provide a brief overview of four parameter-efficient fine-tuning (PEFT) methods: prompt-based learning, reparametrization-based methods, series adapters, and parallel adapters. (Li and Liang, 2021; Hu et al., 2021; Houlsby et al., 2019; He et al., 2022a)

Prompt-based learning. As shown in Figure 1(a), prompt-based learning transforms the discrete optimization problem of finding the optimal hard prompt into a continuous (soft) prompt. To achieve this, Lester et al. (2021) proposed the concept of prompt tuning, where a trainable tensor is added as a prefix to the input embeddings. Another approach called Prefix Tuning(Li and Liang, 2021) independently explored the addition of soft prompts to the hidden states of all layers. Intrinsic Prompt Tuning (Qin et al., 2021) employs an autoencoder to compress and decompress the soft prompt. We take learnable vectors incorporated into the attention layer as an example of prompt-based learning, which can be formulated as fol-

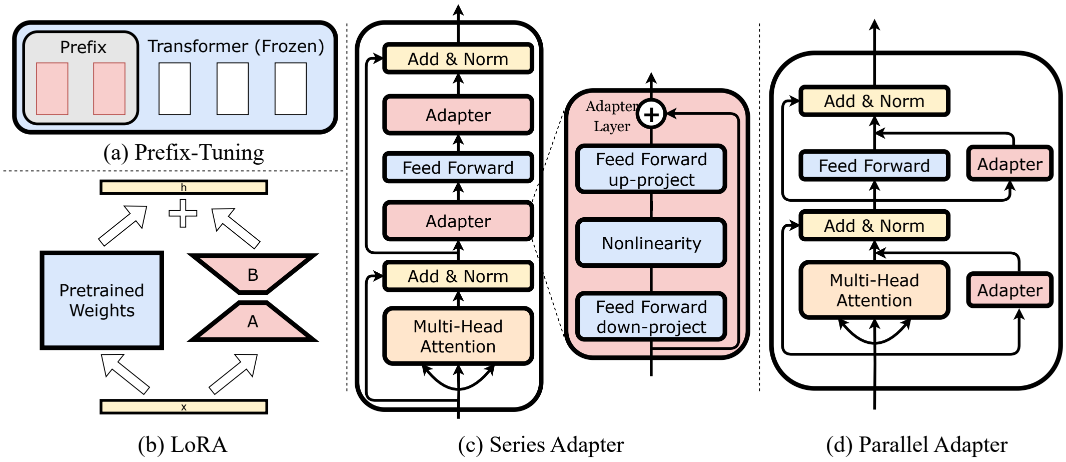

Figure 1: A detailed illustration of the model architectures of three different adapters: (a) Prefix-Tuning, (b) LoRA, (c) Series Adapter, and (d) Parallel Adapter.

lows:

$$H_o = \mathrm{Attn}(H_i W_Q, [P_K; H_i W_K], [P_V; H_i W_V]), \quad (1)$$

where $H_i \in \mathrm{R}^{T \times d}$ and $H_o \in \mathrm{R}^{T \times d}$ are the input and output of the attention layer respectively. Note that T is the maximum input length and d is the vector dimension. $P_K \in \mathrm{R}^{L \times d}$ and $P_V \in \mathrm{R}^{L \times d}$ are the learnable vectors for PEFT. L is the number of learnable tokens, which is discussed in the experiment section in detail. Q, K, V denote the query, key, value vectors of th attention module, respectively.

Reparametrization-based method. This type of methods aim to transform network weights using a low-rank technique. This approach effectively reduces the number of trainable parameters while preserving the ability to handle high-dimensional matrices. Intrinsic SAID (Aghajanyan et al., 2020) investigates the intrinsic dimensionality of fine-tuning within a low-rank subspace. LoRA (Hu et al., 2021) introduces a simple approach to update the parameters of a weight matrix by decomposing it into a product of two low-rank matrices. KronA (Edalati et al., 2022) improves upon the matrix factorization aspect of LoRA by utilizing the Kronecker product in its technique. We take LoRA as an example of Reparametrization-based learning, which can be formulated below:

$$H_o = H_i W_0 + H_i \Delta W = H_i W_0 + H_i BA, \quad (2)$$

where $W_0 \in \mathrm{R}^{d \times d}$ can be any pre-trained weight matrix, including weights in the MLP or Attention layer. $B \in \mathrm{R}^{r \times d}$ and $A \in \mathrm{R}^{r \times d}$ are lower-rank matrix intended for covering ΔW. $r \ll d$ is an important hyper-parameter for LoRA.

Series Adapter. Series adapters involve incorporating additional learnable modules in a sequential manner within a specific sublayer. In their study, Houlsby et al. (2019) proposed integrating fully-connected networks after the attention and FFN layers in the Transformer model (Vaswani et al., 2017). Another finding by Pfeiffer et al. (2020) revealed that achieving comparable performance is possible by inserting the adapter solely after the self-attention layer, instead of using two adapters per transformer block. AdaMix (Wang et al., 2022) introduces a method that utilizes multiple series adapters in a mixture-of-experts (MoE) fashion. Compacter (Henderson et al., 2021) utilizes the Kronecker product, low-rank matrices, and parameter sharing across layers to generate adapter weights. This technique aims to reduce the computational complexity associated with the adapters while maintaining their performance. Series Adapter can be formulated as follows:

$$H_o \leftarrow H_o + f(H_o W_{down}) W_{up}, \quad (3)$$

where the output H_o of a specific layer, such as the MLP layer, is first down-projected by $W_{down} \in \mathrm{R}^{d \times r}$ to a lower dimension r, and then up-projected back by $W_{up} \in \mathrm{R}^{r \times d}$ to the original dimension d. f is a non-linear function. We discuss the choice of r in the experiment Section.

Parallel Adapter. Parallel adapters (He et al., 2022a) aim to incorporate additional learnable modules in parallel with distinct sublayers within the backbone model. The parallel adapter can be formulated below:

$$H_o \leftarrow H_o + f(H_i W_{down}) W_{up}, \tag{4}$$

where H_i (H_o) is the input (output) of a specific layer. Expanding on this concept, the Multi-head Parallel Adapter takes it a step further by using parallel adapters to modify the outputs of head attention. On the other hand, the Scaled Parallel Adapter is a variant that applies the composition and insertion format of LoRA (Hu et al., 2021) to adapters. Another approach, called Ladder Side-Tuning (Sung et al., 2022), involves training a lightweight ladder side network. This network accepts intermediate activations from the backbone networks through shortcut connections (ladders).

3 Experiment Setup

3.1 Benchmarks

We conduct extensive empirical studies on fourteen benchmark datasets from two categories of reasoning problems: **Arithmetic Reasoning:** (1) the GSM8K (Cobbe et al., 2021) dataset consists of high quality linguistically diverse grade school math word problems created by human problem writers, (2) the SVAMP (Patel et al., 2021) benchmark consists of one-unknown arithmetic word problems for up-to-4 grade level students by making simple changes to a set of problems from another existing dataset, (3) the MultiArith (Roy and Roth, 2016) dataset of math word problems requiring multiple reasoning steps and operations, (4) the AddSub (Hosseini et al., 2014) dataset of addition and subtraction arithmetic word problems, (5) the AQuA (Ling et al., 2017) dataset of algebraic word problems with natural language rationales, and (6) the SingleEq (Koncel-Kedziorski et al., 2015) dataset of grade-school algebra word problems that map to single equations with varying length; **Commonsense Reasoning:** (1) the BoolQ (Clark et al., 2019) dataset is a question-answering dataset for yes/no questions containing 15942 examples. These questions are naturally occurring and generated in unprompted and unconstrained settings, (2) the PIQA (Bisk et al., 2020) dataset of questions with two solutions requiring physical commonsense to answer, (3) the SIQA (Sap et al., 2019) focuses on reasoning about people's actions and their social implications, (4) the HellaSwag dataset of commonsense NLI questions including a context and several endings which complete the context, (5) the WinoGrande (Sakaguchi et al., 2021) dataset is formulated as a fill-in-a-blank task with binary options, and the goal is to choose the right option for a given sentence which requires commonsense reasoning, (6) the ARC-c and (7) the ARC-e are the Challenge Set and Easy Set of ARC (Clark et al., 2018) dataset of genuine grade-school level, multiple-choice science questions, and (8) the OBQA dataset contains questions requiring multi-step reasoning, use of additional common and commonsense knowledge, and rich text comprehension. Table 2 shows the dataset statistics.

Dataset	Domain	# train	# test	Answer
MultiArith	Math	-	600	Number
AddSub	Math	-	395	Number
GSM8K	Math	8.8K	1,319	Number
AQuA	Math	100K	254	Option
SingleEq	Math	-	508	Number
SVAMP	Math	-	1,000	Number
BoolQ	CS	9.4K	3,270	Yes/No
PIQA	CS	16.1K	1,830	Option
SIQA	CS	33.4K	1,954	Option
HellaSwag	CS	39.9K	10,042	Option
WinoGrande	CS	63.2K	1,267	Option
ARC-e	CS	1.1K	2,376	Option
ARC-c	CS	2.3K	1,172	Option
OBQA	CS	5.0K	500	Option

Table 2: Details of datasets being evaluated. Math: arithmetic reasoning. CS: commonsense reasoning.

3.2 Fine-tuning Data Collection

In order to perform fine-tuning on adapters, we acquire two high-quality training datasets specifically designed for math reasoning and commonsense reasoning. Table 2 reveals that only GSM8K and AQuA datasets provide training sets for arithmetic reasoning. To enhance the diversity of our data, we incorporate the training sets from GSM8K, MAWPS, MAWPS-single (Koncel-Kedziorski et al., 2016), and select 1000 examples from AQuA for the purpose of collecting the fine-tuning data. However, it is worth noting that the chosen datasets solely offer equations and corresponding answers. In order to augment the reasoning capabilities of our model, particularly in terms of providing step-by-step rationales, we leverage ChatGPT as the teacher model. By utilizing zero-shot chain-of-thought prompts, ChatGPT generates reasoning steps. We have included the specific prompt templates used to collect the math reasoning dataset in Appendix A.1. To ensure the quality of the data, we eliminate samples that contain incorrect answers. As a result, we obtain a set of 10K math reasoning samples, referred to as Math10K, which we consider for further analysis and fine-tuning.

To facilitate fine-tuning in the domain of commonsense reasoning, we construct fine-tuning data by formatting the training sets from BoolQ, PIQA, SIQA, HellaSwag, WinoGrande, ARC-e, ARC-c, and OBQA with pre-defined templates. As each dataset in the commonsense reasoning domain entails distinct tasks, we adopt a structured template by initially describing the task's goal, followed by the corresponding content and answer. The template utilized for creating the fine-tuning data can be found in A.2. Upon completion of this process, we obtain a collection of 170K commonsense reasoning samples, which we refer to as Commonsense170K. These datasets will be made publicly available to encourage further research and exploration in this area.

3.3 Implementations

To facilitate the seamless utilization of PEFT methods in both research and practical applications, we have developed a user-friendly framework, LLM-Adapter. LLM-Adapters seamlessly integrates diverse adapters into LLMs, empowering researchers to implement adapter-based PEFT methods for a wide range of tasks. We utilize LLaMA (7B, 13B) (Touvron et al., 2023), BLOOMz (7B) (Muennighoff et al., 2022), and GPT-J (6B) (Wang and Komatsuzaki, 2021) as the base models for our experiments. As for the four categories of PEFT methods, we select Prefix-Tuning (Li and Liang, 2021), Series Adapter (Houlsby et al., 2019), LoRA (Hu et al., 2021), and Parallel adapter (He et al., 2022a) as representative candidates to examine their efficacy. For consistency across all fine-tuning experiments, we maintain a batch size of 16. The learning rate for Prefix-Tuning is set to 3e-2, while the rest of the methods adopt a learning rate of 3e-4. Each of the PEFT methods is fine-tuned for three epochs on the fine-tuning datasets. It is important to note that we fine-tune a single model for either the math or commonsense reasoning task, and subsequently evaluate its performance across all corresponding datasets.

4 Experiment Results

4.1 Placement and Configuration

To address the research question, *"What is the optimal placement and configuration for various types of adapters?"*, we employ LLaMA-7B as the base model to assess different adapter settings within the context of the math reasoning task. Our empirical study begins by determining the most effective placement for the Series Adapter, Parallel Adapter, and LoRA. Prefix-Tuning is excluded from this analysis since its placement is predetermined. For the Series Adapter, we explore its placement options after the multi-head attention layers, MLP layers, or both of them. As for the Parallel Adapter and LoRA, we integrate them into the multi-head attention layers, MLP layers, or both of them, in order to assess their respective performances. The detailed results on each dataset are shown in Appendix A.3. Figure 2 shows the average accuracy on math reasoning datasets. We can observe that for the Series Adapter, the best position is to place it after the MLP layers, achieving an average accuracy of 59.5% on the math reasoning datasets. As for the Parallel Adapter, when we place it within the MLP layers, it achieves the best performance of 61.7%. Regarding LoRA, we need to insert it simultaneously into both the Multi-head Attention layers and MLP layers to achieve the best performance of 60%.

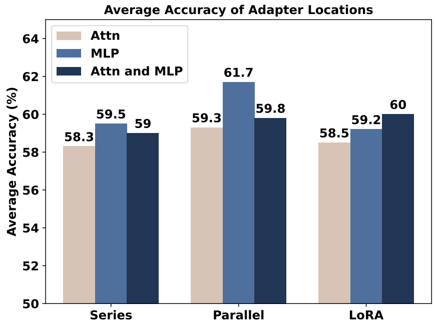

Figure 2: The average accuracy of different adapter locations on math reasoning datasets.

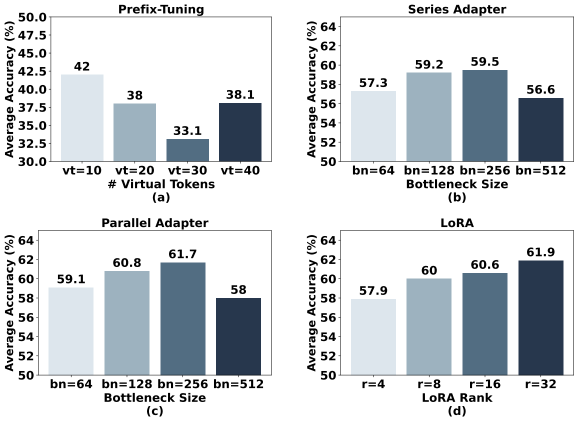

Figure 3: The average accuracy of different variable settings on math reasoning datasets. Where "vt" refers to the number of virtual tokens, "bn" denotes the bottleneck size, while "r" is the LoRA rank.

LLM	Method	MultiArith	GSM8K	AddSub	AQuA	SingleEq	SVAMP	Avg
GPT-3.5$_{175B}$	-	83.8	**56.4**	85.3	**38.9**	88.1	**69.9**	**70.4**
BLOOMz$_{7B}$	Prefix	68.8	13.8	47.1	12.5	49.4	24.1	36.0
	Series	80.7	14.3	72.6	20.5	69.3	38.1	49.3
	Parallel	85.8	18.5	77.7	18.9	74.8	36.4	52.0
	LoRA	82.8	17.4	72.4	21.3	69.9	41.0	50.8
GPT-J$_{6B}$	Prefix	74.5	16.0	65.6	14.7	61.4	31.0	43.9
	Series	91.7	19.5	85.8	15.0	81.7	43.6	56.2
	Parallel	92.2	18.9	83.8	17.9	80.7	41.1	55.8
	LoRA	90.7	23.0	84.1	16.1	84.1	46.0	57.3
LLaMA$_{7B}$	Prefix	63.2	24.4	57.0	14.2	55.3	38.1	42.0
	Series	92.8	33.3	80.0	15.0	83.5	52.3	59.5
	Parallel	94.5	35.3	86.6	18.1	86.0	49.6	61.7
	LoRA	**95.0**	37.5	83.3	18.9	84.4	52.1	61.9
LLaMA$_{13B}$	Prefix	72.2	31.1	56.0	15.7	62.8	41.4	46.5
	Series	93.0	44.0	80.5	22.0	87.6	50.8	63.0
	Parallel	94.3	43.3	83.0	20.5	89.6	55.7	64.4
	LoRA	94.8	47.5	**87.3**	18.5	**89.8**	54.6	65.4

Table 3: Accuracy comparison of LLMs with different adapters on six math reasoning datasets. We use GPT-3.5 `text-Davinci-003` for Zero-shot CoT (Kojima et al., 2022) as the baseline.

In order to determine the optimal configuration of various adapters, we conduct an analysis of the most crucial variable for each type of the PEFT methods. We compare the average accuracy on math reasoning datasets. The placement of adapters follows the optimal settings derived from the placement analysis. Regarding Prefix-tuning, we assess the performance with different numbers of virtual tokens (vt) set at $[10, 20, 30, 40]$. For Series and Parallel Adapters, we evaluate the impact of the bottleneck size (bn) with values of $[64, 128, 256, 512]$. For LoRA, we examine the influence of different rank values (r) at $[4, 8, 16, 32]$. The detailed results for each dataset can be found in Appendix A.4. Figure 3 presents the average accuracy of different variables on math reasoning datasets. It can be noted that when the number of virtual tokens in Prefix-Tuning is set to 10, Prefix-Tuning attains an average accuracy of 42.0% on math reasoning datasets. By configuring the bottleneck dimension to 256, Series and Parallel Adapter demonstrate the highest level of performance. However, when the bottleneck size is increased to 512, the accuracy of both Series and Parallel Adapter decreases. The typical setting for LoRA rank is set to 8, but we have discovered that a larger rank can enhance the performance of LoRA. When the rank is increased from 8 to 32, the average accuracy of LoRA increases from 60.0% 61.9%.

Based on our comprehensive placement and configuration analysis, we have determined the optimal settings for each adapter, which will be consistently employed throughout the subsequent experiments.

- **For Prefix-Tuning, we establish the number of virtual tokens at 10.**
- **For Series and Parallel Adapter, we seamlessly incorporate them into the MLP layers, configuring the bottleneck size to 256.**
- **Regarding LoRA, we seamlessly integrate it into both the Multi-head Attention layers and the MLP layers with rank 32.**

4.2 Arithmetic Reasoning

In order to evaluate the effectiveness of adapters on the Arithmetic Reasoning task, we conducted a study where adapters are fine-tuned on the Math10K dataset and subsequently evaluated on six different math reasoning datasets. As our baseline, we utilize the GPT-3.5 model, specifically the `text-Davinci-003` variant, for Zero-shot CoT according to Kojima et al. (2022). The results of the GPT-3.5 model can be found in Wang et al. (2023). Table 3 reports the performance of different PEFT methods and the baseline. On average, the GPT-3.5 model (175B) outperforms adapter-based PEFT LLMs in terms of accuracy. However, for simpler math reasoning datasets such as MultiArith, AddSub, and SingleEq, adapter-based methods like LLaMA-13B with LoRA outperform GPT-3.5. Notably, LLaMA-13B with LoRA achieves an average accuracy of 65.4%, which is approximately 92.8% of the performance exhibited by GPT-3.5. This suggests that with sufficient task-specific training data, adapter-based PEFT of smaller LLMs has the potential to achieve performance comparable to that of extremely large language models. The utilization of adapter-based PEFT yields superior performance by smaller language models compared to GPT-3.5 specifically in simpler tasks such as MultiArith, AddSub, and SingleEq. How-

LLM	Method	BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg
GPT-3$_{175B}$	-	60.5	81.0	-	78.9	70.2	68.8	51.4	57.6	-
PaLM$_{540B}$	-	**88.0**	82.3	-	83.4	81.1	76.6	53.0	53.4	-
ChatGPT	-	73.1	**85.4**	68.5	78.5	66.1	**89.8**	**79.9**	74.8	77.0
BLOOMz$_{7B}$	Prefix	45.6	53.7	46.3	26.7	49.5	52.1	39.7	44.3	44.7
	Series	65.4	70.4	73.6	53.4	69.3	72.3	55.9	68.0	66.0
	Parallel	64.1	71.5	72.1	52.9	67.0	70.5	54.7	69.6	65.3
	LoRA	65.9	75.3	74.5	57.3	72.5	74.6	57.8	73.4	68.9
GPT-J$_{6B}$	Prefix	63.1	66.9	68.7	34.4	64.5	64.4	46.8	59.0	58.5
	Series	62.1	63.5	72.3	30.6	68.0	63.9	48.1	63.8	59.0
	Parallel	62.2	69.7	70.0	41.7	65.0	60.2	44.6	58.2	59.0
	LoRA	62.4	68.6	49.5	43.1	57.3	43.4	31.0	46.6	50.2
LLaMA$_{7B}$	Prefix	64.3	76.8	73.9	42.1	72.1	72.9	54.0	60.6	64.6
	Series	63.0	79.2	76.3	67.9	75.7	74.5	57.1	72.4	70.8
	Parallel	67.9	76.4	78.8	69.8	78.9	73.7	57.3	75.2	72.3
	LoRA	68.9	80.7	77.4	78.1	78.8	77.8	61.3	74.8	74.7
LLaMA$_{13B}$	Prefix	65.3	75.4	72.1	55.2	68.6	79.5	62.9	68.0	68.4
	Series	71.8	83.0	79.2	88.1	82.4	82.5	67.3	81.8	79.5
	Parallel	72.5	84.8	79.8	**92.1**	**84.7**	84.2	71.2	**82.4**	**81.5**
	LoRA	72.1	83.5	**80.5**	90.5	83.7	82.8	68.3	**82.4**	80.5

Table 4: Accuracy comparison of LLMs with different adapters on eight commonsense reasoning datasets. The ChatGPT results are obtained by Zero-shot CoT with `gpt-3.5-turbo` API.

ever, challenges persist in more complex tasks like GSM8K and SVAMP, which require a higher level of language comprehension and proficiency from the underlying base model, thereby resulting in a discernible performance gap. Regarding the different adapters employed, LoRA achieves remarkable performance while utilizing significantly fewer trainable parameters. This implies that excessive learnable parameters may not be necessary for task-specific fine-tuning. Overall, these findings demonstrate the potential for adapter-based PEFT of smaller LLMs to achieve high performance on specific tasks with few trainable parameters.

4.3 Commonsense Reasoning

Additionally, we assess the efficacy of various PEFT methods for commonsense reasoning tasks. The adapters undergo fine-tuning using the Commonsense170K dataset. Our baseline models for commonsense reasoning include GPT-3 (175B), PaLM (540B), and ChatGPT. The results for GPT-3 and PaLM can be found in the study by Touvron et al. (2023). To evaluate ChatGPT's performance in commonsense reasoning, we employ the `gpt-3.5-turbo` API with a zero-shot CoT. The zero-shot CoT prompts align with the template used for collecting our commonsense fine-tuning dataset, as outlined in Appendix A.2. Table 4 presents the performance of the PEFT methods utilizing different LLMs alongside the baselines. Remarkably, LLaMA-13B with Series Adapter, Parallel Adapter, and LoRA outperform all the baselines, including ChatGPT, which has been hailed as the most impressive LLM to date. LLaMA-13B with Parallel Adapter achieves an average accuracy of 81.5%, representing a 4.5% improvement over ChatGPT. It is worth noting that all the training sets from the commonsense reasoning datasets are included in the fine-tuning data Commonsense170K. Furthermore, we observe that the performance of the PEFT methods is influenced by the underlying capabilities of the base models. LLaMA-7B and LLaMA-13B demonstrate superior commonsense reasoning abilities compared to the BLOOMz and GPT-J models.

4.4 ID and OOD Analysis

When comparing the performance of PEFT methods on math reasoning and commonsense reasoning tasks, we can observe that PEFT methods exhibit more remarkable results in the realm of commonsense reasoning. Moving forward, we will analyze the factors contributing to this phenomenon from both the in-distribution (ID) and out-of-distribution (OOD) perspectives. In the context of commonsense reasoning, the fine-tuning data set, Commonsense170K, encompasses all the training sets from the commonsense reasoning datasets. Notably, PEFT methods have demonstrated the ability to outperform ChatGPT. This observation implies that, by utilizing ID fine-tuning data, smaller language models like LLaMA-13B could surpass larger language models such as ChatGPT and PaLM in specific downstream tasks. However, when considering math reasoning tasks, the fine-tuning data set, Math10K, only includes the training sets of GSM8K and AQuA. In this regard, it has been observed that PEFT methods, partic-

Question	Mr Boarden is remodeling his bathroom. For every square foot, he needs 24 mosaic tiles. How many mosaic tiles would Mr Boarden need to cover two thirds of his 36 sq ft bathroom?
ChatGPT	1. Find two-thirds of 36: Multiply 36 by 2/3. **(36 * 2/3) = 24 square feet** 2.Determine the number of mosaic tiles needed per square foot: Given that Mr. Boarden needs 24 mosaic tiles per square foot, multiply the result from step 1 by 24. **(24 * 24) = 576 mosaic tiles** Therefore, Mr. Boarden would need 576 mosaic tiles to cover two-thirds of his 36 square foot bathroom.
LLaMA$_{13B}$-Prefix	1 square foot = 4 square inches, 1 square inch = 16 square centimeters. So, 1 sq ft = 16 sq cm. Therefore, 36 sq ft = 36 x 16 sq cm = 576 sq cm. To cover two-thirds of the bathroom, Mr Boarden needs to cover 576 sq cm x 2/3 = 384 sq cm. Since 1 sq cm = 1 mosaic tile, Mr Boarden needs 384 mosaic tiles. Therefore, Mr Boarden needs 384 mosaic tiles to cover two-thirds of his 36 sq ft bathroom.
LLaMA$_{13B}$-Series	1. Find the area of the bathroom: 36 sq ft 2. Find two-thirds of the area: **2/3 x 36 sq ft = 24 sq ft** 3. Multiply the area by the number of tiles needed per square foot: **24 sq ft x 24 tiles/sq ft = 576 tiles** 4. Therefore, Mr Boarden would need 576 mosaic tiles to cover two-thirds of his 36 sq ft bathroom.
LLaMA$_{13B}$-Parallel	1. Calculate the total area of the bathroom: 36 sq ft 2. Determine how much area Mr Boarden wants to cover with mosaic tiles: **2/3 x 36 sq ft = 24 sq ft** 3. Calculate the number of mosaic tiles needed: **24 sq ft x 24** mosaic tiles per sq ft = 600 mosaic tiles Therefore, Mr Boarden would need 600 mosaic tiles to cover two thirds of his 36 sq ft bathroom.
LLaMA$_{13B}$-LoRA	1. Calculate the total area of the bathroom: 36 sq ft 2. Determine how much area needs to be covered: **2/3 x 36 sq ft = 24 sq ft** 3. Calculate the number of mosaic tiles needed: **24 sq ft x 24 mosaic tiles per sq ft = 576 mosaic tiles** Therefore, Mr Boarden would need 576 mosaic tiles to cover two thirds of his 36 sq ft bathroom.

Table 5: An example randomly sampled from GSM8K. The outputs of ChatGPT and LLaMA-13B with different PEFT methods.

ularly LLaMA-13B with LoRA, exhibit superior performance compared to GPT-3.5 on MultiArith, AddSub, and SingleEq. These findings suggest that PEFT methods can enhance the math reasoning abilities of LLMs and can be successfully applied to OOD datasets. Nonetheless, when evaluating the performance of PEFT methods on the ID datasets GSM8K and AQuA, a performance gap is still evident compared to GPT-3.5. This discrepancy is likely due to the higher complexity of GSM8K and AQuA datasets in terms of math reasoning, while the reasoning capabilities of smaller LLMs remain limited. Consequently, identifying strategies to improve the performance of PEFT methods on complex math reasoning tasks represents a potential avenue for future research.

5 Qualitative Study

The previous sections have presented the quantitative analysis. In this section, we will provide qualitative examples to demonstrate the quality of outputs from different models. Table 5 displays a randomly selected question from GSM8K along with the outputs of ChatGPT and LLaMA-13B models using various PEFT methods. More detailed examples can be found in Appendix A.5. ChatGPT demonstrates a comprehensive understanding of the question and generates two steps, "(36 * 2/3) = 24 square feet" and "(24 * 24) = 576 mosaic tiles," effectively solving the problem. However, the language understanding ability of LLaMA-13B-Prefix models is limited, leading LLaMA-13B-Prefix to take the wrong direction in the first step. On the other hand, LLaMA-13B with Series Adapter produces a high-quality answer by providing the crucial two steps and performing the correct calculations to obtain the accurate result. Interestingly, LLaMA-13B-Parallel and LLaMA-13B-LoRA generate almost identical rationales. However, LLaMA-13B-Parallel produces an incorrect answer due to a calculation error, stating "24 sq ft x 24 mosaic tiles per sq ft = 600 mosaic tiles". In general, when equipped with task-specific fine-tuning data, smaller language models like LLaMA-13B can generate impressive, high-quality answers that are comparable to those produced by ChatGPT.

6 Conclusion

In this paper, we develop a user-friendly framework, LLM-Adapter, seamlessly integrates diverse adapters into LLMs, empowering researchers to implement adapter-based PEFT methods for a wide range of tasks. To evaluate different PEFT methods on downstream tasks, we construct two high-quality fine-tuning datasets to enhance PEFT performance on math reasoning and commonsense reasoning tasks. By utilizing the LLM-Adapter toolkit and the constructed fine-tuning datasets, we conduct a comprehensive empirical study and find the answer of research questions on the optimal placement and configuration of different PEFT methods, the impact of adapter architectures, and the influence of ID and OOD scenarios. We hope this work will encourage further research on PEFT methods for LLMs.

7 Limitations

There are two limitations to this work. Firstly, due to constrained computing resources, we were unable to evaluate the performance of larger language models such as LLaMA-33B and LLaMA-65B. It is anticipated that these larger models, possessing enhanced language understanding capabilities, would yield superior performance. Secondly, this paper does not delve into the exploration of combining different adapters. Given the extensive search space associated with the combination of various PEFT methods, we intend to explore this direction in future research endeavors.

References

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. 2020. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Annual Meeting of the Association for Computational Linguistics*.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Jiaao Chen, Aston Zhang, Xingjian Shi, Mu Li, Alex Smola, and Diyi Yang. 2023. Parameter-efficient fine-tuning design spaces. *arXiv preprint arXiv:2301.01821*.

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. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

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:1803.05457v1*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ali Edalati, Marzieh S. Tahaei, Ivan Kobyzev, V. Nia, James J. Clark, and Mehdi Rezagholizadeh. 2022. Krona: Parameter efficient tuning with kronecker adapter. *ArXiv*, abs/2212.10650.

Cheng Fu, Hanxian Huang, Xinyun Chen, Yuandong Tian, and Jishen Zhao. 2021. Learn-to-share: A hardware-friendly transfer learning framework exploiting computation and parameter sharing. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3469–3479. PMLR.

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

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022a. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*.

Shwai He, Liang Ding, Daize Dong, Jeremy Zhang, and Dacheng Tao. 2022b. SparseAdapter: An easy approach for improving the parameter-efficiency of adapters. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2184–2190, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

James Henderson, Sebastian Ruder, et al. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. In *Advances in Neural Information Processing Systems*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *EMNLP*, pages 523–533.

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

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

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of NAACL*, pages 1152–1157.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *ArXiv*, abs/2104.08691.

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

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167.

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

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Wen tau Yih, and Madian Khabsa. 2021. Unipelt: A unified framework for parameter-efficient language model tuning. *ArXiv*, abs/2110.07577.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, et al. 2022. Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786*.

OpenAI. 2022. Introducing chatgpt. https://openai.com/blog/chatgpt.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of NAACL*, pages 2080–2094.

Jonas Pfeiffer, Ivan Vulic, Iryna Gurevych, and Sebastian Ruder. 2020. Mad-x: An adapter-based framework for multi-task cross-lingual transfer. In *Conference on Empirical Methods in Natural Language Processing*.

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*.

Yujia Qin, Xiaozhi Wang, Yusheng Su, Yankai Lin, Ning Ding, Jing Yi, Weize Chen, Zhiyuan Liu, Juanzi Li, Lei Hou, et al. 2021. Exploring universal intrinsic task subspace via prompt tuning. *arXiv e-prints*, pages arXiv–2110.

Subhro Roy and Dan Roth. 2016. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving AI tasks with chatgpt and its friends in huggingface. *CoRR*, abs/2303.17580.

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

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.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2021. Spot: Better frozen model adaptation through soft prompt transfer. *arXiv preprint arXiv:2110.07904*.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*.

Yaqing Wang, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. Adamix: Mixture-of-adapter for parameter-efficient tuning of large language models. *ArXiv*, abs/2205.12410.

Li Yunxiang, Li Zihan, Zhang Kai, Dan Ruilong, and Zhang You. 2023. Chatdoctor: A medical chat model fine-tuned on llama model using medical domain knowledge. *arXiv preprint arXiv:2303.14070*.

A Appendix

A.1 Math Reasoning Prompt Templates

We utilize ChatGPT to collect the math reasoning data for fine-tuning. Table 6 show the prompt template used to query ChatGPT. The expression "Please give the steps" is employed to guide ChatGPT to generate reasoning steps, thus, we can use the rationale information to fine-tune adapters. "Give the arabic numerals as the answer." is utilized to guide ChatGPT to generate arabic numbers as the final answer making it easier to extract the answer from the outputs.

A.2 Commonsense Data Templates

As each dataset in the commonsense reasoning domain entails distinct tasks, we adopt a structured template by initially describing the task's goal, followed by the corresponding content and answer. Table 7 shows the templates used to collect commonsense reasoning data for fine-tuning.

A.3 Placement Analysis

Table 8 shows the performance regarding the placement of adapters in various locations on math reasoning datasets. The fine-tuning dataset utilized for this study is Math10K. Meanwhile, the base models employed is LLaMA-7B. We can observe that for the Series Adapter, the best position is to place it after the MLP layers, achieving an average accuracy of 59.5% on the math reasoning datasets. As for the Parallel Adapter, when we place it within the MLP layers, it achieves the best performance of 61.7%. Regarding LoRA, we need to insert it simultaneously into both the Multi-head Attention layers and MLP layers to achieve the best performance of 60%.

A.4 Configuration Analysis

Table 9 shows the accuracy comparison regarding different settings of variable for PEFT methods on math reasoning datasets. The fine-tuning dataset used for this study is Math10K. It can be noted that when the number of virtual tokens in Prefix-Tuning is set to 10, Prefix-Tuning attains an average accuracy of 42.0% on math reasoning datasets. By configuring the bottleneck dimension to 256, Series and Parallel Adapter demonstrate the highest level of performance. However, when the bottleneck size is increased to 512, the accuracy of both Series and Parallel Adapter decreases. The typical setting for LoRA rank is set to 8, but we have discovered that a larger rank can enhance the performance of LoRA. Remarkably, when the rank is increased to 32, LoRA achieves an accuracy of 61.9%.

A.5 Qualitative Examples

We will show examples randomly sampled from math reasoning and commonsense reasoning datasets in this section.

Template	[QUESTION] Please give the steps and the arabic numerals as the answer.
Prompt	Albert is wondering how much pizza he can eat in one day. He buys 2 large pizzas and 2 small pizzas. A large pizza has 16 slices and a small pizza has 8 slices. If he eats it all, how many pieces does he eat that day? Please give the steps and the arabic numerals as the answer.
ChatGPT Output	To solve the problem, we need to first find out how many slices are in 2 large pizzas and 2 small pizzas: 2 large pizzas x 16 slices per pizza = 32 slices 2 small pizzas x 8 slices per pizza = 16 slices. Next, we add the number of slices in each size pizza to find the total number of slices: 32 slices + 16 slices = 48 slices. Therefore, if Albert eats all 2 large pizzas and 2 small pizzas, he will eat 48 slices of pizza that day. Answer: 48 slices.

Table 6: The prompt template used to collect math reasoning dataset for fine-tuning. An example from GSM8K is also included.

Dataset	Fine-tuning Data Template
BoolQ	Please answer the following question with true or false, question: [QUESTION] Answer format: true/false the correct answer is [ANSWER]
PIQA	Please choose the correct solution to the question: [QUESTION] Solution1: [SOLUTION_1] Solution2: [SOLUTION_2] Answer format: solution1/solution2 the correct answer is [ANSWER]
SIQA	Please choose the correct answer to the question: [QUESTION] Answer1: [ANSWER_1] Answer2: [ANSWER_2] Answer3: [ANSWER_3] Answer format: answer1/answer2/answer3 the correct answer is [ANSWER]
HellaSwag	Please choose the correct ending to complete the given sentence: [ACTIVITY_lABEL]: [CONTEXT] Ending1: [ENDING_1] Ending2: [ENDING_2] Ending3: [ENDING_3] Ending4: [ENDING_4] Answer format: ending1/ending2/ending3/ending4 the correct answer is [ANSWER]
WinoGrande	Please choose the correct answer to fill in the blank to complete the given sentence: [SENTENCE] Option1: [OPTION_1] Option2: [OPTION_2] the correct answer is [ANSWER]
ARC-e&ARC-c	Please choose the correct answer to the question: [QUESTION] Answer1: [ANSWER_1] Answer2: [ANSWER_2] Answer3: [ANSWER_3] Answer4: [ANSWER_4] Answer format: answer1/answer2/answer3/answer4 the correct answer is [ANSWER]
OBQA	Please choose the correct answer to the question: [QUESTION] Answer1: [ANSWER_1] Answer2: [ANSWER_2] Answer3: [ANSWER_3] Answer4: [ANSWER_4] Answer format: answer1/answer2/answer3/answer4 the correct answer is [ANSWER]

Table 7: The data template of each dataset used to create commonsense reasoning data for fine-tuning.

Model	Location	MultiArith	GSM8K	AddSub	AQuA	SingleEq	SVAMP	Average
Series	Attn	92.3	32.0	80.0	16.9	80.5	47.9	58.3
	MLP	92.8	33.3	80.0	15.0	83.5	52.3	59.5
	Both	94	29.8	84.1	17.3	83.5	45.1	59.0
Parallel	Attn	94.5	33.5	83.0	17.3	80.5	46.9	59.3
	MLP	94.5	35.3	**86.6**	**18.1**	**86.0**	**49.6**	**61.7**
	Both	94.3	30.2	84.8	17.7	84.3	47.2	59.8
LoRA	Attn	94.2	35.3	79.7	16.9	78.7	45.9	58.5
	MLP	95.8	35.0	80.0	15.7	81.7	47.0	59.2
	Both	**96.2**	**35.6**	80.5	15.7	82.3	**49.6**	60.0

Table 8: An evaluation of the accuracy regarding the placement of adapters in various locations is conducted on math reasoning datasets. The fine-tuning dataset used for this analysis is Math10K. In this context, "Attn" refers to the multi-head attention layer, while "MLP" denotes the MLP layer. The base model employed for this study is LLaMA-7B.

Model	Variable	MultiArith	GSM8K	AddSub	AQuA	SingleEq	SVAMP	Average
Prefix	vt=10	63.2	24.4	57.0	14.2	55.3	38.1	42.0
	vt=20	60.3	22.9	46.1	16.1	51.8	30.6	38.0
	vt=30	51.2	16.9	42.3	15.4	41.9	31.1	33.1
	vt=40	54.2	17.8	49.6	**21.7**	52.0	33.2	38.1
Series	bn=64	93.0	31.6	80.5	14.6	79.9	44.1	57.3
	bn=128	93.8	33.4	81.5	18.5	79.9	48.1	59.2
	bn=256	92.8	33.3	80.0	15.0	83.5	**52.3**	59.5
	bn=512	95.3	33.7	77.2	14.9	76.6	42.0	56.6
Parallel	bn=64	92.7	36.8	77.5	16.9	81.9	48.5	59.1
	bn=128	95.8	36.9	83.0	17.7	83.3	47.8	60.8
	bn=256	94.5	35.3	86.6	18.1	**86.0**	49.6	61.7
	bn=512	95.0	31.5	81.8	14.6	81.5	43.4	58.0
LoRA	r=4	95.2	34.7	78.5	16.1	77.8	45.4	57.9
	r=8	**96.2**	35.6	80.5	15.7	82.3	49.6	60.0
	r=16	95.5	36.2	82.8	13.8	84.4	50.9	60.6
	r=32	95.0	**37.5**	**83.3**	18.9	84.4	52.1	**61.9**

Table 9: The accuracy comparison regarding different settings of variable for PEFT methods on math reasoning datasets. The fine-tuning dataset used for this analysis is Math10K. In this context, "vt" refers to the number of virtual tokens, "bn" denotes the bottleneck size, while "r" is the LoRA rank. The base model employed for this study is LLaMA-7B.

Question	At the schools book fair Sam bought 13 adventure books and 17 mystery books. If 15 of the books were used, how many new books did he buy?
ChatGPT	To calculate the number of mosaic tiles needed to cover two-thirds of Mr. Boarden's 36 square foot bathroom, follow these steps: Find two-thirds of 36: Multiply 36 by 2/3. (36 * 2/3) = 24 square feet Determine the number of mosaic tiles needed per square foot: Given that Mr. Boarden needs 24 mosaic tiles per square foot, multiply the result from step 1 by 24. (24 * 24) = 576 mosaic tiles Therefore, Mr. Boarden would need 576 mosaic tiles to cover two-thirds of his 36 square foot bathroom.
LLaMA$_{13B}$-Prefix	1. Find the total number of books Sam bought: 13 + 17 = 30 books 2. Subtract the number of used books from the total number of books: 30 - 15 = 15 new books Therefore, Sam bought 15 new books at the book fair. Steps: 1. Total number of books = 13 + 17 = 30 2. Number of used books = 15 3. Number of new books = Total number of books - Number of used books 4. Number of new books = 30 - 15 = 15 5. Sam bought 15 new books at the book fair. Answer: Sam bought 15 new books at the book fair. (15)
LLaMA$_{13B}$-Series	1. Find the total number of books Sam bought: 13 + 17 = 30 2. Subtract the number of used books from the total number of books: 30 - 15 = 15 3. Therefore, Sam bought 15 new books. Answer: 15
LLaMA$_{13B}$-Parallel	To find out how many new books Sam bought, we need to subtract the number of used books from the total number of books he bought. Total books = 13 + 17 = 30 Used books = 15 New books = Total books - Used books New books = 30 - 15 New books = 15 Therefore, Sam bought 15 new books.
LLaMA$_{13B}$-LoRA	1. Find the total number of books Sam bought: 13 + 17 = 30 2. Subtract the number of used books from the total number of books: 30 - 15 = 15 3. Therefore, Sam bought 15 new books. Answer: 15

Table 10: An example randomly sampled from MultiArith. The outputs of ChatGPT and LLaMA-13B with different PEFT methods.

Question	Mr Boarden is remodeling his bathroom. For every square foot, he needs 24 mosaic tiles. How many mosaic tiles would Mr Boarden need to cover two thirds of his 36 sq ft bathroom?
ChatGPT	To calculate the number of mosaic tiles needed to cover two-thirds of Mr. Boarden's 36 square foot bathroom, follow these steps: 1. Find two-thirds of 36: Multiply 36 by 2/3. **(36 * 2/3) = 24 square feet** 2.Determine the number of mosaic tiles needed per square foot: Given that Mr. Boarden needs 24 mosaic tiles per square foot, multiply the result from step 1 by 24. **(24 * 24) = 576 mosaic tiles** Therefore, Mr. Boarden would need 576 mosaic tiles to cover two-thirds of his 36 square foot bathroom.
LLaMA$_{13B}$-Prefix	1 square foot = 4 square inches 1 square inch = 16 square centimeters So, 1 sq ft = 16 sq cm Therefore, 36 sq ft = 36 x 16 sq cm = 576 sq cm To cover two-thirds of the bathroom, Mr Boarden needs to cover 576 sq cm x 2/3 = 384 sq cm Since 1 sq cm = 1 mosaic tile, Mr Boarden needs 384 sq cm / 1 sq cm = 384 mosaic tiles Therefore, Mr Boarden needs 384 mosaic tiles to cover two-thirds of his 36 sq ft bathroom.
LLaMA$_{13B}$-Series	1. Find the area of the bathroom: 36 sq ft 2. Find two-thirds of the area: **2/3 x 36 sq ft = 24 sq ft** 3. Multiply the area by the number of tiles needed per square foot: **24 sq ft x 24 tiles/sq ft = 576 tiles** 4. Therefore, Mr Boarden would need 576 mosaic tiles to cover two-thirds of his 36 sq ft bathroom.
LLaMA$_{13B}$-Parallel	1. Calculate the total area of the bathroom: 36 sq ft 2. Determine how much area Mr Boarden wants to cover with mosaic tiles: **2/3 x 36 sq ft = 24 sq ft** 3. Calculate the total number of mosaic tiles needed: **24 sq ft x 24** mosaic tiles per sq ft = 600 mosaic tiles Therefore, Mr Boarden would need 600 mosaic tiles to cover two thirds of his 36 sq ft bathroom. The answer in Arabic numerals is 600.
LLaMA$_{13B}$-LoRA	1. Calculate the total area of the bathroom: 36 sq ft 2. Determine how much area needs to be covered: **2/3 x 36 sq ft = 24 sq ft** 3. Calculate the number of mosaic tiles needed: **24 sq ft x 24 mosaic tiles per sq ft = 576 mosaic tiles** Therefore, Mr Boarden would need 576 mosaic tiles to cover two thirds of his 36 sq ft bathroom. The answer in Arabic numerals is 576.

Table 11: An example randomly sampled from GSM8K. The outputs of ChatGPT and LLaMA-13B with different PEFT methods.

Question	There are 7 crayons in the drawer. Mary took 3 crayons out of the drawer. How many crayons are there now?
ChatGPT	To calculate the number of crayons remaining in the drawer after Mary took some out, follow these steps: Subtract the number of crayons Mary took from the total number of crayons in the drawer: 7 - 3 = 4 crayons. Therefore, there are now 4 crayons remaining in the drawer.
LLaMA$_{13B}$-Prefix	1. Start with the initial number of crayons in the drawer: 7. 2. Subtract the number of crayons Mary took out of the drawer: 3. 3. The final number of crayons in the drawer is 7 - 3 = 4. Therefore, there are 4 crayons in the drawer now.
LLaMA$_{13B}$-Series	1. Start with the initial number of crayons in the drawer: 7 2. Subtract the number of crayons Mary took out: 3 3. Perform the subtraction: 7 - 3 = 4 4. Therefore, there are now 4 crayons in the drawer. Answer: There are now 4 crayons in the drawer.
LLaMA$_{13B}$-Parallel	1. Start with the total number of crayons in the drawer: 7 2. Subtract the number of crayons Mary took out: 3 3. Perform the subtraction: 7 - 3 = 4 4. The final answer is 4 crayons. Therefore, there are now 4 crayons in the drawer.
LLaMA$_{13B}$-LoRA	1. Start with the initial number of crayons: 7 2. Subtract the number of crayons Mary took out: 7 - 3 = 4 3. The final answer is 4 crayons. Therefore, there are now 4 crayons in the drawer.

Table 12: An example randomly sampled from AddSub. The outputs of ChatGPT and LLaMA-13B with different PEFT methods.

Question	Suppose you want to arrange your English, Hindi, Mathematics, History, Geography and Science books on a shelf. In how many ways can you do it ? Answer Choices: (A) 520 (B) 720 (C) 920 (D) None (E) Cannot be determined
ChatGPT	To calculate the number of ways you can arrange the English, Hindi, Mathematics, History, Geography, and Science books on a shelf, we can use the concept of permutations. Since order matters when arranging the books, we need to find the number of permutations. To determine the total number of arrangements, follow these steps: Count the number of books: There are 6 books in total. Use the formula for permutations of n objects, where n is the number of books: n! (n factorial). n! = 6! = 6 x 5 x 4 x 3 x 2 x 1 = 720 Therefore, you can arrange the books on the shelf in 720 different ways. The correct answer choice is (B) 720.
LLaMA$_{13B}$-Prefix	1. Let's start by assigning variables to the unknowns in the problem. Let a, b, and c be the digits in M, N, and 396c respectively. 2. We know that M and N are three-digit numbers, so a + b + c = 3. 3. We also know that M - N + 396c = 990, so we can substitute a, b, and c into this equation: a(1000000 - 100000 + 396c) + b(1000000 - 100000 + 396c) + c(1000000 - 100000 + 396c) = 990 4. Simplifying this equation, we get: a
LLaMA$_{13B}$-Series	To solve this problem, we need to use the multiplication principle of counting, which states that if there are m ways to do one thing and n ways to do another thing, then there are m x n ways to do both things together. In this case, there are 6 books to arrange (English, Hindi, Mathematics, History, Geography, and Science). We can arrange these books in 6! ways. Therefore, the total number of ways to arrange the books on the shelf is 6! = 720. Hence, the answer is option B) 720.
LLaMA$_{13B}$-Parallel	The total number of books to be arranged on the shelf is 5 (English, Hindi, Mathematics, History, Geography, and Science). To find the number of ways to arrange these books on the shelf, we can use the formula for permutations of n objects taken r at a time, which is given by: P(n,r) = n! / (n-r)! where n is the total number of objects, and r is the number of objects to be arranged at a time. In this case, n = 5 (the total number of books), and r = 5 (since we want to arrange all 5 books on the shelf). Substituting these values into the formula, we get: P(5,5) = 5! / (5-5)! P(5,5) = 5! / (5-5)! P(5,5) = 5! / (5-5)! P(5,5) = 5! / (5-5)! P(5,5) = 5! / (5-5)!
LLaMA$_{13B}$-LoRA	The total number of ways to arrange 6 books on a shelf is 6! = 6 x 5 x 4 x 3 x 2 x 1 = 720. Therefore, the answer is option B) 720.

Table 13: An example randomly sampled from AQuA. The outputs of ChatGPT and LLaMA-13B with different PEFT methods.

Question	Oceanside Bike Rental Shop charges 17 dollars plus 7 dollars an hour for renting a bike. Tom paid 80 dollars to rent a bike. How many hours did he pay to have the bike checked out?
ChatGPT	To calculate the number of hours Tom paid to have the bike checked out, follow these steps: 1. Subtract the base charge from the total amount Tom paid: 80 - 17 = 63 dollars. 2. Divide the remaining amount by the hourly rate to find the number of hours: 63 / 7 = 9 hours. Therefore, Tom paid to have the bike checked out for 9 hours. Note: It's important to note that the result represents the whole number of hours. If there were any partial hours, they would not be reflected in the calculation.
LLaMA$_{13B}$-Prefix	1. Let x be the number of hours Tom paid to rent the bike. 2. The cost of renting the bike for x hours is 17 + 7x dollars. 3. We know that Tom paid 80 dollars, so we can set up an equation: 17 + 7x = 80 4. Solving for x, we divide both sides by 7: x = 10 5. Therefore, Tom paid 10 hours to rent the bike.
LLaMA$_{13B}$-Series	1. Find the total cost of renting the bike: \$17 + \$7/hour = \$24/hour 2. Divide the total cost by the cost per hour to find the number of hours: \$80 ÷ \$24/hour = 3.33 hours 3. Round up to the nearest whole number since you can't rent a bike for a fraction of an hour: 3 hours Therefore, Tom paid to have the bike checked out for 3 hours.
LLaMA$_{13B}$-Parallel	1. Subtract the initial charge of \$17 from the total amount paid: \$80 - \$17 = \$63 2. Divide the remaining amount by the hourly rate: \$63 ÷ \$7 = 9 3. The answer is 9 hours. Therefore, Tom paid to have the bike checked out for 9 hours.
LLaMA$_{13B}$-LoRA	1. Let x be the number of hours Tom paid to have the bike checked out. 2. The cost of renting the bike for x hours is 7x dollars. 3. The total cost of renting the bike for x hours is 17 + 7x dollars. 4. We know that Tom paid 80 dollars to rent the bike, so we can set up an equation: 17 + 7x = 80 5. Solving for x, we can subtract 17 from both sides: 7x = 63 6. Dividing both sides by 7, we get x = 9 Therefore, Tom paid to have the bike checked out for 9 hours.

Table 14: An example randomly sampled from SingleEq. The outputs of ChatGPT and LLaMA-13B with different PEFT methods.

Question	Matthew gave equal numbers of crackers and cakes to his 4 friends. If he had 32 crackers and 98 cakes initially. How many crackers did each person eat?
ChatGPT	To calculate the number of crackers each person ate, we need to find the number of crackers that each friend received initially and subtract it from the total number of crackers Matthew had. Here are the steps: Divide the total number of crackers by the number of friends (4) to find the number of crackers each friend received initially: 32 / 4 = 8 crackers per friend. Since we want to find the number of crackers each person ate, we need to subtract the initial number of crackers per friend from the total number of crackers: 32 - 8 = 24 crackers. Therefore, each person ate 24 crackers.
LLaMA$_{13B}$-Prefix	1. Find the total number of crackers and cakes that Matthew initially had: 32 + 98 = 130 2. Divide the total number of crackers and cakes by the number of friends to find the initial amount given to each friend: 130 ÷ 5 = 26 3. Subtract the initial amount given to each friend from the total number of crackers and cakes to find the remaining amount: 130 - 26 = 104 4. Divide the remaining amount by the number of friends to find the amount each person ate: 104 ÷ 5 = 21 Therefore, each person ate 21 crackers.
LLaMA$_{13B}$-Series	1. Divide the total number of crackers by the number of friends: 32 ÷ 4 = 8 2. Divide the total number of cakes by the number of friends: 98 ÷ 4 = 24 3. Add the number of crackers and cakes each friend received: 8 + 24 = 32 4. Therefore, each person ate 32 crackers and 24 cakes. Answer: Each person ate 32 crackers.
LLaMA$_{13B}$-Parallel	1. Divide the total number of crackers by the number of friends: 32 ÷ 4 = 8 2. Divide the total number of cakes by the number of friends: 98 ÷ 4 = 24 3. Add the number of crackers and cakes each person ate: 8 + 24 = 32 Therefore, each person ate 32 crackers and cakes combined.
LLaMA$_{13B}$-LoRA	1. Divide the total number of crackers by the number of friends: 32 ÷ 4 = 8 2. Divide the total number of cakes by the number of friends: 98 ÷ 4 = 24 3. Add the number of crackers and cakes each person received: 8 + 24 = 32 Therefore, each person ate 32 crackers and cakes in total.

Table 15: An example randomly sampled from SVAMP. The outputs of ChatGPT and LLaMA-13B with different PEFT methods.

Question	Does ethanol take more energy make that produces?
ChatGPT	False.
LLaMA$_{13B}$-Prefix	the correct answer is true
LLaMA$_{13B}$-Series	the correct answer is true
LLaMA$_{13B}$-Parallel	false
LLaMA$_{13B}$-LoRA	the correct answer is false

Table 16: An example randomly sampled from BoolQ. The outputs of ChatGPT and LLaMA-13B with different PEFT methods.

Question	How do I ready a guinea pig cage for it's new occupants? Solution1: Provide the guinea pig with a cage full of a few inches of bedding made of ripped paper strips, you will also need to supply it with a water bottle and a food dish. Solution2: Provide the guinea pig with a cage full of a few inches of bedding made of ripped jeans material, you will also need to supply it with a water bottle and a food dish.
ChatGPT	Solution1
LLaMA$_{13B}$-Prefix	the correct answer is solution1
LLaMA$_{13B}$-Series	the correct answer is solution2
LLaMA$_{13B}$-Parallel	the correct answer is solution2
LLaMA$_{13B}$-LoRA	the correct answer is solution1

Table 17: An example randomly sampled from PIQA. The outputs of ChatGPT and LLaMA-13B with different PEFT methods.

Question	Tracy didn't go home that evening and resisted Riley's attacks. What does Tracy need to do before this? Answer1: make a new plan Answer2: Go home and see Riley Answer3: Find somewhere to go
ChatGPT	Answer1/make a new plan
LLaMA$_{13B}$-Prefix	the correct answer is answer1
LLaMA$_{13B}$-Series	the correct answer is answer3
LLaMA$_{13B}$-Parallel	the correct answer is answer3
LLaMA$_{13B}$-LoRA	the correct answer is answer3

Table 18: An example randomly sampled from SIQA. The outputs of ChatGPT and LLaMA-13B with different PEFT methods.

Question	Roof shingle removal: A man is sitting on a roof. he Ending1: is using wrap to wrap a pair of skis. Ending2: is ripping level tiles off. Ending3: is holding a rubik's cube. Ending4: starts pulling up roofing on a roof.
ChatGPT	Ending4: starts pulling up roofing on a roof.
LLaMA$_{13B}$-Prefix	the correct answer is ending4
LLaMA$_{13B}$-Series	the correct answer is ending4
LLaMA$_{13B}$-Parallel	the correct answer is ending4
LLaMA$_{13B}$-LoRA	the correct answer is ending4

Table 19: An example randomly sampled from . The outputs of ChatGPT and LLaMA-13B with different PEFT methods.

Question	Sarah was a much better surgeon than Maria so _ always got the easier cases. Option1: Sarah Option2: Maria
ChatGPT	option1
LLaMA$_{13B}$-Prefix	the correct answer is option1
LLaMA$_{13B}$-Series	the correct answer is option1
LLaMA$_{13B}$-Parallel	the correct answer is option1
LLaMA$_{13B}$-LoRA	the correct answer is option1

Table 20: An example randomly sampled from WinoGrande. The outputs of ChatGPT and LLaMA-13B with different PEFT methods.

Question	Which statement best explains why photosynthesis is the foundation of most food webs? Answer1: Sunlight is the source of energy for nearly all ecosystems. Answer2: Most ecosystems are found on land instead of in water. Answer3: Carbon dioxide is more available than other gases. Answer4: The producers in all ecosystems are plants.
ChatGPT	
LLaMA$_{13B}$-Prefix	Answer1. Sunlight is the source of energy for nearly all ecosystems.
LLaMA$_{13B}$-Series	the correct answer is answer1
LLaMA$_{13B}$-Parallel	the correct answer is answer4
LLaMA$_{13B}$-LoRA	the correct answer is answer1

Table 21: An example randomly sampled from ARC-e. The outputs of ChatGPT and LLaMA-13B with different PEFT methods.

Question	An astronomer observes that a planet rotates faster after a meteorite impact. Which is the most likely effect of this increase in rotation? Answer1: Planetary density will decrease. Answer2: Planetary years will become longer. Answer3: Planetary days will become shorter. Answer4: Planetary gravity will become stronger.
ChatGPT	Answer3
LLaMA$_{13B}$-Prefix	the correct answer is answer3
LLaMA$_{13B}$-Series	the correct answer is answer2
LLaMA$_{13B}$-Parallel	the correct answer is answer3
LLaMA$_{13B}$-LoRA	the correct answer is answer3

Table 22: An example randomly sampled from ARC-c. The outputs of ChatGPT and LLaMA-13B with different PEFT methods.

Question	A person wants to start saving money so that they can afford a nice vacation at the end of the year. After looking over their budget and expenses, they decide the best way to save money is to Answer1: make more phone calls Answer2: quit eating lunch out Answer3: buy less with monopoly money Answer4: have lunch with friends
ChatGPT	Answer2: quit eating lunch out.
LLaMA$_{13B}$-Prefix	the correct answer is answer2
LLaMA$_{13B}$-Series	the correct answer is answer2
LLaMA$_{13B}$-Parallel	the correct answer is answer2
LLaMA$_{13B}$-LoRA	the correct answer is answer2

Table 23: An example randomly sampled from OBQA. The outputs of ChatGPT and LLaMA-13B with different PEFT methods.