# PARA: Parameter-Efficient Fine-tuning with Prompt Aware Representation Adjustment

**Zequan Liu**[1*] **Yi Zhao**[2*] **Ming Tan**[3] **Wei Zhu**[4†] **Aaron Xuxiang Tian** [5]
[1] RWTH Aachen University, Aachen, Germany
[2] University of Pennsylvania, USA
[3] Southern University of Science and Technology, Shenzhen, China
[4] University of Hong Kong, Hong Kong, China
[5] Carnegie Mellon University, Pittsburgh, USA

## Abstract

In the realm of parameter-efficient fine-tuning (PEFT) methods, while options like LoRA are available, there is a persistent demand in the industry for a PEFT approach that excels in both efficiency and performance within the context of single-backbone multi-tenant applications. This paper introduces a new and straightforward PEFT technique, termed Prompt Aware Representation Adjustment (PARA). The core of our proposal is to integrate a lightweight vector generator within each Transformer layer. This generator produces vectors that are responsive to input prompts, thereby adjusting the hidden representations accordingly. Our extensive experimentation across diverse tasks has yielded promising results. Firstly, the PARA method has been shown to surpass current PEFT benchmarks in terms of performance, despite having a similar number of adjustable parameters. Secondly, it has proven to be more efficient than LoRA in the single-backbone multi-tenant scenario, highlighting its significant potential for industrial adoption.

## 1 Introduction

In industrial applications, large language models (LLMs) are frequently utilized in a single-instance, multi-tenant configuration, as highlighted in Chen et al.'s 2023 study on PunicaML (Chen et al., 2023). An instance of this is when an LLM vendor offers a model as a service (MaaS), as described by Gan et al. in 2023 (Gan et al., 2023). In this arrangement, various clients can tailor the LLM to their specific needs using their own parameter-efficient fine-tuning (PEFT) modules. A locally installed LLM is typically required to manage a variety of tasks for different tenants, each with their own set of PEFT parameters. However, while techniques like Low-Rank Adaptation (LoRA) (Hu et al., 2021) are adept at fine-tuning LLMs, they add considerable latency to each generation step because the low-rank components cannot be integrated into the main model structure. On the other hand, $(IA)^3$ (Liu et al., 2022a), which relies solely on dot product operations, is a more efficient PEFT approach but may lack the necessary expressiveness. Consequently, there is a pressing need in the industry for a PEFT method that strikes a balance between efficiency and effectiveness.

In this work, we propose a novel PEFT method called Prompt Aware Representation Adjustment (PARA) (depicted in Figure 1). Our method fine-tuned the LLMs by directly modifying the hidden representations in the model by multiplying them by adjusting vectors and thus regulating the behaviors of LLMs. Unlike the previous literature like Liu et al. (2022a) or Ben-Zaken et al. (2021), we introduce a novel prompt-aware mechanism to the PEFT method. The adjusting vectors are not randomly initialized and fixed across different input prompts. Instead, we install a vector generator (VG) before each Transformer layer, taking the input prompts' hidden states as input and generating the adjusting vectors as outputs. VG is a lightweight bottleneck architecture consisting of a pooling layer, a down-projection layer, an activation function, and an up-projection layer.

Certainly! Here's the revised version of your text with the LaTeX formatting preserved:

We perform a wide range of experiments across a diverse set of tasks to establish the efficacy of our PARA approach. It's important to note that our approach consistently surpasses robust PEFT benchmarks with similar adjustable parameter limits, particularly the latest LoRA iterations, $(IA)^3$ (Liu et al., 2022a), and BitFit (Ben-Zaken et al., 2021). We also demonstrate that our method exhibits substantially reduced latency in a multi-tenant environment compared to LoRA-based approaches, highlighting its suitability for real-world applications.

*Equal contributions.
† Corresponding author. Email: michael-wzhu91@gmail.com.



*Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 728–737
November 12-16, 2024 

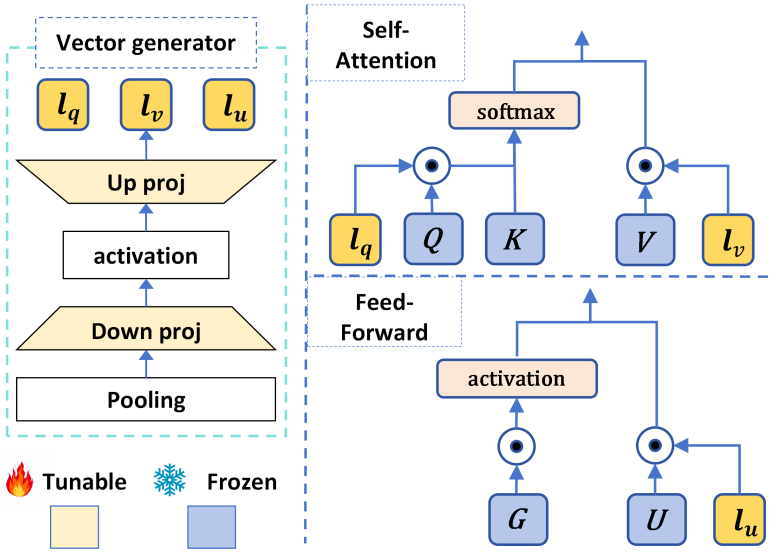


Figure 1: A schematic representation of our PARA approach is depicted below. On the left, the vector generator is composed of several components, including a pooler, a down-projection layer, an activation function, and an up-projection layer. This generator takes the hidden states of the prompt as input and produces adjusting vectors as output. On the right, these adjusting vectors are used to scale the Query (Q) and Value (V) hidden states within the MHSA (Multi-Head Self-Attention) module, as well as the Up (U) hidden states within the feed-forward network.

Our contributions can be encapsulated as follows:

- We introduce an innovative PEFT technique, PARA, which refines LLMs by producing adjustment vectors based on input prompts to alter the hidden states of LLMs.
- Our comprehensive experiments and analyses reveal that our PARA system is (a) feasible and surpasses the competition within comparable parameter constraints. (b) swift during the inference phase for LLMs.

## 2 Related works

Parameter-efficient fine-tuning (PEFT) entails selectively optimizing a subset of parameters within a large pre-trained model while leaving the core model architecture intact for adaptation purposes (Ding et al., 2022; Zhang et al., 2023b). In contrast, addition-based techniques involve integrating extra neural components or parameters into the existing model framework. Notable contributions in this domain include 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., 2022b), $(IA)^3$ (Liu et al., 2022a), and BitFit (Ben-Zaken et al., 2021). Conversely, specification-based methods involve the explicit designation of parameters that are either adjustable or subject to pruning (Ben-Zaken et al., 2021; Guo et al., 2021; Zhao et al., 2020). The reparameterization-based strategies have garnered significant interest (Hu et al., 2021). These approaches convert the parameters being optimized into a format that is both low-rank and parameter-efficient. Such PEFT methods are underpinned by the insight that the dimensionality intrinsic to fine-tuning is relatively low (Aghajanyan et al., 2021). LoRA (Hu et al., 2021), for instance, posits that the variation in weights during tuning is characterized by a low intrinsic rank, and thus focuses on optimizing the low-rank factorization of the weight matrix changes. PEFT techniques have found broad application, particularly with the rise of open-source large-scale language models (Zhao et al., 2023) and the trend of tailoring these models to specific use cases through instruction tuning (Taori et al., 2023; Dettmers et al., 2023).

In this research, we introduce a novel framework known as PARA, which is designed for the parameter-efficient fine-tuning of Large Language Models (LLMs). This approach not only enhances efficiency during LLM inference but also delivers superior performance across various downstream applications.

## 3 Methods

### 3.1 Preliminaries

**Transformer model** Currently, the most widely used open-sourced large language models adopt the stacked Transformer architecture (Vaswani et al., 2017). The transformer block is primarily constructed using two key submodules: a multi-head self-attention (MHA) layer and a fully connected feed-forward (FFN) layer. Denote the input sequence's length as $l$, the hidden states' dimension as $d_{model}$, and the dimension at the FFN module as $d_{ffn}$. The MHA is given as follows:[1]

$$\text{softmax}\left(\frac{QK}{\sqrt{d_{model}}}\right)V, \tag{1}$$

where $Q = xW^Q$, $K = xW^K$, $V = xW^V$, $x \in \mathbf{R}^{l \times d_{model}}$ is the input tensor. $W^Q, W^K, W^V \in \mathbf{R}^{d_{model} \times d_{model}}$ are the query, key, and value projection layers (denoted as the Query, Key, and Value modules, or the Q, K, V modules). The FFN module consists of linear transformations and an activation function $g^{ffn}$ such as ReLU or GELU (Hendrycks and Gimpel, 2016). Take the FFN module in the LlaMA-2 models (Touvron et al., 2023) as example:

$$(g^{ffn}(G) * U)W^D, \tag{2}$$

where $G = xW^G$, $U = xW^U$, $W^G, W^U \in \mathbf{R}^{d_{model} \times d_{ffn}}$ (denoted as Gate and Up module, or the G and U modules).

**Task formulation** Denote the task's training set as $\mathcal{D}_{\text{train}} = (x_m, y_m), m = 1, 2, ..., M$, where $M$ represents the number of samples. In this work, we only consider the case where input $x_m$ and target $y_m$ are both text sequences.

### 3.2 PARA

Now we present the framework of our novel Prompt Aware Representation Adjustment (PARA) method.

**Formulation** Denote the hidden state of the input prompt with length $T_{ins}$ at the current Transformer layer as $\mathbf{h}$. As shown in Figure 1, the vector generator VG() use $\mathbf{h}$ as input to generate three learned vectors, $l_q, l_v \in \mathbb{R}^{d_{model}}$ and $l_u \in \mathbb{R}^{d_{ffn}}$, with a vector generator:

$$l_q,\ l_v,\ l_u = \text{VG}(\mathbf{h}), \tag{3}$$

and these generated vectors are used to modify the hidden representations in the self-attention and FFN modules. Thus, under PARA, the self-attention mechanism of Transformer (in Equation 1) is changed to

$$\text{softmax}\left(Q^{'}K/\sqrt{d_{model}}\right)V^{'}, \tag{4}$$

where $Q^{'} = l_q \odot Q, V^{'} = l_v \odot V$, and $\odot$ denotes the element-wise dot product. And the FFN module (Equation 2) is modified to

$$(g^{ffn}(G) \odot U^{'})W^D, \tag{5}$$

where $U^{'} = l_u \odot U$.[2][3]

**Vector generator** Now we introduce the central component of our PARA framework, the vector generator denoted as VG(). This function accepts $\mathbf{h}$ as its input and is composed of a pooling module along with a pair of projection operations, each accompanied by an activation function. The process begins by converting $\mathbf{h}$ into a single vector using the Pooler() function. In line with the works of Radford et al. (2018) and Lewis et al. (2019), Pooler() outputs the vector representation corresponding to the final token in the prompt. Subsequently, the pooled vector is projected from the dimension $d_{model}$ down to $r < d_{model}$ through a projection layer defined by $W^{vg}_{down} \in \mathbb{R}^{d_{model} \times r}$. This is followed by the application of an activation function $g^{vg}$, after which the vector is projected to the dimension $d_{out} = 2 * d_{model} + d_{ffn}$ via another projection layer, utilizing the weight matrix $W^{vg}_{up}$ and bias term $b^{vg}_{up}$. Mathematically, the vector generator can be expressed by the following equations:

$$\begin{aligned} l &= (g^{vg}(\text{Pooler}(\mathbf{h})W^{vg}_{down}))W^{vg}_{up} + b^{vg}_{up}, \\ l_q,\ l_v,\ l_u &= \text{Split}(l), \end{aligned} \tag{6}$$

where the Split() function is responsible for dividing the vector into three separate vectors, each of dimension $d_{model}$, $d_{model}$, and $d_{ffn}$, respectively.

The concept of prompt-awareness is derived from studies on in-context learning. As shown by Rubin et al. (2022) and Li et al. (2023), enhancing the performance of Large Language Models (LLMs) can be achieved by dynamically creating

[1] We omit the multi-head setting for simplicity of illustrations.

[2] We use the "broadcasting notation" in the Equations 4 and 5. Take so that the $(m, n)$-th entry of $U^{'}$ is $l_u[n] \odot U[m, n]$.

[3] From our preliminary experiments, we find that generating adjustment vectors for the other hidden states like K and G will not result in clear performance gains.

an expanded prompt that includes examples tailored to the specific input prompt. It has been observed that distinct input prompts necessitate unique examples to evoke more effective responses from LLMs. Similarly, the idea of tailoring PEFT parameters to the input prompt could enhance the method's expressive capabilities and more precisely control the conduct of LLMs.

It's important to recognize that causal language models (CLM), which are based on decoders, often utilize the KV cache mechanism[4] to enhance efficiency during the generation process. The vector generators in our system integrate flawlessly with this KV cache mechanism. This is because the vectors $l_q$, $l_v$, and $l_u$ are produced when the input instruction (or prompt) is initially processed by the LLM. These vectors are then reused for the generation of subsequent tokens, and the vector generators are not invoked again. On the other hand, the LoRA method introduces reparameterizations to the model's parameters, necessitating that its low-rank weight matrices be included in the forward calculations for each token generation step, which results in increased latency.

## 4 Experiments

In this section, we conduct experiments to evaluate our PARA method.

### 4.1 Baselines

We compare our PARA framework with the current SOTA PEFT baseline methods: (a) (IA)$^3$ (Liu et al., 2022a), which multiplies learnable vectors to the hidden representations of LLMs. (b) Houlsby-Adapter (Houlsby et al., 2019). (c) Learned-Adapter (Zhang et al., 2023b). (d) LoRA (Hu et al., 2021). (e) AdaLoRA (Zhang et al., 2023a). (f) SSP (Hu et al., 2022), which combines different PEFT methods. The baselines are implemented using Transformers (Wolf et al., 2020a) or their open-sourced codes.

### 4.2 Datasets and evaluation metrics

We experiment on the following benchmark tasks: (a) three benchmark question-answering tasks: SQuAD (Rajpurkar et al., 2016) and two tasks from the SuperGLUE benchmark (Wang et al., 2019) (BoolQ, COPA). (b) two widely used LLM evaluation benchmarks, MT-Bench (Zheng et al., 2023), MMLU (Hendrycks et al., 2020). (c) A proprietary LLM evaluation benchmark, LLM-Eval1, for internal LLM developments of an industrial participant. (d) a proprietary high-school-level mathematical solving dataset, HSM10K. (e) a proprietary SQL generation task, Q2SQL. The above tasks' dataset introductions, statistics, and evaluation metrics are detailed in Appendix A.

### 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 LLM, LlaMA-2 7B released by Meta (Touvron et al., 2023) as the pretrained backbone model. We will also use the LlaMA-2 13B model and Gemma 2B (Team et al., 2024) in the ablation studies.

**Prediction heads** After receiving a prompt or instruction, all the predictions are generated using the language modeling head (LM head). For decoding during inference, we use beam search with beam size 3.

**Hyper-parameters for the PARA framework** In our experiments, unless otherwise specified, we set: (a) the bottleneck dimension $r$ of the PARA vector generator to 12, (b) the activation function $g^{vg}$ to the GeLU activation function (Hendrycks and Gimpel, 2016). (c) The $W^{vg}_{down}$ is initialized with a Gaussian distribution of mean 0 and std 0.02. $W^{vg}_{up}$ is zero initialized, and $b^{vg}_{up}$ is initialized with ones. Under the above settings, our PARA method will introduce 8.9M tunable parameters to LlaMA-2 7B.

**Training settings for PARA** Utilizing the HuggingFace Transformers (Wolf et al., 2020b), PEFT (Mangrulkar et al., 2022), or the original code repositories, we implement all the methods for training and prediction tasks. When fine-tuning the LlaMA-2 7B model, the sequence length is capped at 2048. The training epochs are limited to a maximum of 10. The batch size is adjusted to 16 for tasks with fewer than 10k training samples, and 128 for larger datasets. AdamW serves as the optimizer, employing a linear learning rate decay strategy with a 6% warm-up period over the training steps. The learning rate is configured at 1e-4. All other hyper-parameters align with those used by Wolf et al. (2020b). The model's performance is assessed on the development set every 200 steps. Early stopping is initiated with a pa-

[4] https://www.dipkumar.dev/becoming-the-unbeatable/posts/gpt-kvcache/

| Datasets | #train | #dev | #test | $\|\mathcal{Y}\|$ | Type | Labels | Metrics |
|---|---|---|---|---|---|---|---|
| BoolQ | 9.4k | 1.6k | 1.6k | 2 | Question Answering | True, False | acc |
| COPA | 0.4k | 0.05k | 0.05k | 2 | Question Answering | choice1, choice2 | acc |
| SQuAD | 87k | 1k | 5.9k | - | Question Answering | - | f1-em |
| MT-Bench | - | - | 80 | - | Question Answering | - | GPT-4 scores |
| MMLU | - | 1.5k | 14.1k | - | Question Answering | - | acc |
| HSM10K | 9K | 0.6K | 0.7K | - | Math reasoning | - | acc |
| Q2SQL | 60k | 4K | 10K | - | SQL generation | - | acc |
| LLM-Eval1 | - | - | 3.6k | - | Question Answering | - | acc |
| UltraChat | 766k | 7.7k | - | - | Instruction tuning | - | - |

Table 1: The statistics of the datasets evaluated in this work. $|\mathcal{Y}|$ is the number of classes for a classification task.

| **Method** | **Tunable Params** | **HSM10K (acc)** | **Q2SQL (acc)** | **SQuAD (f1-em)** | **BoolQ (acc)** | **COPA (acc)** |
|---|---|---|---|---|---|---|
| Full-FT | 7B | 57.9 | 82.9 | 89.5 | 88.7 | 91.9 |
| ***Baselines PEFT methods*** | | | | | | |
| Housbly-Adapter | 9.4M | 52.8 | 80.4 | 87.3 | 84.5 | 90.4 |
| Learned-Adapter | 9.5M | 53.7 | 81.3 | 87.6 | 85.9 | 90.6 |
| SSP | 8.6M | 54.6 | 81.5 | 87.4 | 86.4 | 91.1 |
| $(IA)^3$ | 9.8M | 54.3 | 81.2 | 87.6 | 86.2 | 90.7 |
| LoRA | 10.0M | 55.1 | 81.8 | <u>87.7</u> | 86.3 | 90.9 |
| AdaLoRA | 10.0M | <u>55.6</u> | <u>82.2</u> | 87.5 | <u>87.0</u> | <u>91.2</u> |
| ***Our proposed method*** | | | | | | |
| PARA | 8.9M | **56.3** | **82.8** | **88.5** | **87.7** | **92.0** |

Table 2: The Overall comparison of the SQuAD, BoolQ, COPA, HSM10K and Q2SQL tasks. The backbone model is LLM-Assist 7B. We report the median performance over five random seeds. Bold and Underline indicate the best and the second-best results. The metric for each task is explained in Appendix A.2.

tience level of 10, meaning training will be halted if the model fails to record a lower loss on the development set for 10 consecutive evaluations. The optimal checkpoint identified on the development set is then applied to make predictions on the test set.

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

### 4.4 Main results

The outcomes of our experiments on the SQuAD, BoolQ, COPA, HSM10K, and Q2SQL benchmarks are detailed in Table 2, where the count of adjustable parameters is listed in the second column. The data in Table 2 indicates that our PARA approach surpasses the standard methods on all five benchmarks, with an equivalent or reduced number of adjustable parameters. Notably, PARA achieves better results than the previous state-of-the-art LoRA-style baselines, namely LoRA and AdaLoRA, while using a similar parameter count.

After fine-tuning the LLM-Assist 7B model on the UltraChat dataset (Ding et al., 2023) using our PARA configuration or the AdaLoRA techniques, we proceed to assess its performance on the demanding benchmarks: MT-Bench, MMLU, and LLM-Eval1. The trials are executed in a zero-shot scenario, with no exemplar instances appended to the input prompts. The outcomes are detailed in Table 3. Aligning with the findings from the prior experiments (Table 2), our PARA approach surpasses the AdaLoRA techniques across the three benchmarks, indicating that PARA is more effective in bolstering the directive tuning proficiency of expansive language models.

### 4.5 Further analysis

**Analysis of the inference efficiency** To showcase the inference efficiency of our PARA approach, we proceed to juxtapose the GPU memory usage and the rate of generation for PARA, LoRA,

| Method | MT-Bench gpt4-score (↑) | MMLU acc | LLM-Eval1 acc |
|---|---|---|---|
| AdaLoRA | 7.13 | 46.5 | 56.8 |
| PARA | 7.21 | 47.4 | 57.7 |

Table 3: Performance of general-purpose instruction tuning using the PARA and AdaLoRA methods. The backbone model is LLM-Assist 7B. ↑ means the metric is higher the better.

| Method | Beam size | Speed (tps) | Memory cost (MiB) |
|---|---|---|---|
| LoRA | 1 | 25.1 | 14616 |
| | 3 | 21.9 | 16104 |
| $(IA)^3$ | 1 | 33.1 | 14572 |
| | 3 | 27.6 | 16036 |
| PARA | 1 | 32.8 | 14512 |
| | 3 | 27.6 | 15986 |

Table 4: The memory and speed of LlaMA-2 7B for generating responses with different PEFT methods.

and $(IA)^3$. In the course of this experiment, parameters of LoRA have not been integrated into the main model to emulate a single-LLM multi-tenant configuration as indicated in (Chen et al., 2023). We have capped the creation of new tokens to 32, utilizing beam search with a beam width of either 1 or 3. The initial instruction's length is set at 274, employing the LlaMA-2 tokenizer. We execute the generation process a total of 100 instances to ascertain the average metric estimates, thereby diminishing the element of randomness. We introduce two key metrics for gauging efficiency: (a) the apex memory expenditure during the generation phase, and (b) the rate of token generation per second (tps). The comparative data is delineated in Table 4.

As depicted in Table 4, it is evident that: (a) our PARA approach possesses a similar number of adjustable parameters, memory usage, and generation rate to $(IA)^3$. (b) PARA outperforms LoRA in terms of speed. The enhanced velocity of PARA over LoRA can be attributed to several elements: (i) our vector generation process is both minimal and efficient during the inference phase. (ii) The vectors, $l_q$, $l_v$, $l_u$, are generated solely upon the input of instructions to the LLM and prior to the creation of the initial new token. These vectors are then reused in subsequent generation stages with the aid of KV-cache, eliminating the need for repeated invocation of the vector generators. Conversely, the LoRA technique necessitates the model to engage the LoRA modules at every generation stage, leading to increased latency.

**Ablation on the pretrained backbones** Our principal experiments were carried out utilizing the LlaMA-2 7B model. In order to showcase the versatility of our approach, additional experiments have been executed on both the LlaMA-2 13B model and the Gemma 2B model. The corresponding outcomes are detailed within Table 5. Furthermore, our approach surpasses the performance of the foundational methodologies on these alternative model architectures.

## 5 Conclusion

This study introduces PARA, an innovative approach for the parameter-efficient fine-tuning of expansive language models. We integrate a vector generator within each Transformer layer to produce adjustment vectors that modulate the functionality of the LLM core. The vector generator utilizes the hidden states of the input prompts as inputs and features a lightweight bottleneck design. PARA offers greater efficiency in inference compared to LoRA, as it operates harmoniously with the KV-cache system. Our experiments across a range of tasks show that PARA surpasses the performance of standard methods while maintaining high inference efficiency. PARA is advantageous for industrial applications that leverage LLMs.

| Method | BoolQ (acc) | SQuAD (f1-em) |
|---|---|---|
| ***Results for LlaMA-2 13B model*** | | |
| $(IA)^3$ | 89.6 | 90.6 |
| LoRA | 90.0 | 90.9 |
| AdaLoRA | 90.2 | 91.6 |
| PARA | **90.9** | **92.1** |
| ***Results for Gemma 2B*** | | |
| $(IA)^3$ | 82.7 | 78.1 |
| LoRA | 82.8 | 78.4 |
| AdaLoRA | 83.0 | 78.8 |
| PARA | **83.6** | **79.7** |

Table 5: Results for different PEFT methods on the BoolQ and SQuAD benchmarks. The backbone LMs are LlaMA-2 13B and Gemma 2B. The metrics are explained in Appendix A.2.

## Limitations

We showed that our proposed method can greatly improve the performance of parameter-efficient tuning on diverse tasks and different pretrained models (i.e., LlaMA-2 7B, LlaMA-2 13B model and Gemma 2B), while maintaining efficiency during inference. 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 experimented. 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 broader types of tasks. And we will explore it in future work.

## Ethics Statement

The finding and proposed method aims to improve the parameter-efficient tuning in terms of performance and 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, 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 testing 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.

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.

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, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3029–3051.

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.

Wensheng Gan, Shicheng Wan, and Philip S. Yu. 2023. Model-as-a-service (maas): A survey. *2023 IEEE International Conference on Big Data (BigData)*, pages 4636–4645.

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.

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*.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv: Learning*.

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

Edward J Hu, 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.

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

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

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

Xiaonan Li, Kai Lv, Hang Yan, Tianyang Lin, Wei Zhu, Yuan Ni, Guotong Xie, Xiaoling Wang, and Xipeng Qiu. 2023. Unified demonstration retriever for in-context learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4644–4668.

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.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022b. 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`.

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

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. *OpenAI*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671.

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*.

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.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *ArXiv*, abs/1905.00537.

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.

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.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, and Yoav Artzi. 2020. Revisiting few-sample bert fine-tuning. *ArXiv*, abs/2006.05987.

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.

# A Appendix for the datsets and evaluation metrics

## A.1 Datasets

We now introduce the datasets we used for experiments. The detailed statistics of these tasks are presented in Table 1.

**COPA & BoolQ** These two tasks are question answering tasks in the format of binary choices, and are included in the SuperGLUE benchmark. Since the original test sets are not publicly available for these tasks, we follow Zhang et al. (2020); Mahabadi et al. (2021) to divide the original validation set in half, using one half for validation and the other for testing.

**SQuAD task** Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage, or the question might be unanswerable. This task is one of the most widely studied question answering task in the field. In this work, we use the v1.1 version of SQuAD. Since the original test sets are not publicly available for these tasks, we follow Zhang et al. (2020); Mahabadi et al. (2021) and split 1k samples from the training set as the development set, and use the original development

set as the test set. The detailed statistics of this task is presented in Table 1.

**HSM10K benchmark** HSM10K is a dataset of 10.3K high quality high school level problems created by the math teachers. These problems are the most difficult ones from a wide source of math tests. The solving steps are generated by GPT-4 and then checked/rewritten by math teachers to ensure accuracy. We use this dataset to improve the math reasoning abilities of LLMs. The dataset is split into 9k/0.6K/0.7K train/dev/test sets.

**Q2SQL dataset** Q2SQL consists of a corpus of 74K hand-annotated SQL query and natural language question pairs. This proprietary dataset is collected from a company in the health insurance company, where the SQL are primarily related to analyzing insurance policies. These SQL queries are further split into training (60k examples), development (4k examples) and test sets (10k examples). In this work, we will ask the LLMs to generate SQL queries based on the given natural language questions.

**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.

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

**The LLM-Eval1 benchmark** This benchmark is a proprietary dataset, designated to challenge the LLMs for reasoning, world knowledge, and task solving. This dataset is used internally to facilitate LLM development. LLM-Eval1 contains a suite of 47 challenging tasks from multiple domains including literature, healthcare, security, coding assistant, and software development and testing. The number of test samples are 3,569.

**The UltraChat dataset** UltraChat (Ding et al., 2023) is an open-source, large-scale, and multi-round dialogue data curated with the help of OpenAI's GPT-3-Turbo API. To ensure generation quality, two separate GPT-3-Turbo APIs are adopted in generation, where one plays the role of the user to generate queries and the other generates the response. The user model is carefully prompted to mimic human user behavior and the two APIs are called iteratively to create a dialogue. There are 774k dialogues in the dataset, and we split it into a 99:1 train/validate set for the FanLoRA workflow.

### A.2 Evaluation metrics/protocols

For the BoolQ and COPA tasks, we report accuracy following (Wang et al., 2019).

For the SQuAD dataset, we also report the average of the F1 score and the exact match score (denoted as f1-em).

For the HSM10K task, we will consider the correctness of the final answers. Thus, we report accuracy (denoted as acc).

For the Q2SQL, we will consider the correctness of the generated SQL queries. A predicted SQL query is correct if and only if it can be executed and obtains the same results with the ground truth.

For the MMLU and LLM-Eval1 tasks, we will directly consider the correctness of the final answers. Thus, we report accuracy (denoted as acc).

For evaluating the quality of instruction tuned LLMs, we follow the practice of utilizing GPT-4 as a unbiased reviewer (Zheng et al., 2023). 80 instructions from the MT-Bench is set as a test set. 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 AdaLoRA and FanLoRA'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.