# FanLoRA: Fantastic LoRAs and Where to Find Them in Large Language Model Fine-tuning

**Aaron Xuxiang Tian**[1*] **Yi Zhao**[2*] **Congrui Yin**[3] **Wei Zhu**[4†]
**Xing Tian**[4] **Yi Ge**[4]
[1] Carnegie Mellon University, Pittsburgh, USA
[2] University of Pennsylvania, USA
[3] University of Minnesota, USA
[4] University of Hong Kong, Hong Kong, China

## Abstract

Full-parameter fine-tuning is computationally prohibitive for large language models (LLMs), making parameter-efficient fine-tuning (PEFT) methods like low-rank adaptation (LoRA) increasingly popular. However, LoRA and its existing variants introduce significant latency in multi-tenant settings, hindering their applications in the industry. To address this issue, we propose the Fantastic LoRA (FanLoRA) framework, which consists of four steps: (a) adding LoRA modules to all the Transformer linear weights and fine-tuning on a large-scale instruction tuning dataset. (b) The importance of each module is then assessed using a novel importance scoring method. (c) only the most critical modules per layer are retained, resulting in the FanLoRA setting. (d) The FanLoRA setting is applied to fine-tune various downstream tasks. Our extensive experiments demonstrate that: (a) FanLoRA outperforms existing PEFT baselines across a wide collection of tasks with comparable tunable parameters. (b) FanLoRA significantly reduces the inference latency of LoRA, making it valuable for further broadening the applications of LLMs in the industry.

## 1 Introduction

In the era of large language models (LLMs), parameter-efficient fine-tuning (PEFT) (Zhang et al., 2023b; Zhao et al., 2023) has raised much attention in the research field since in PEFT, the tunable parameters are often less than 1% of the LLMs and the hardware requirements for fine-tuning are significantly decreased. Among many PEFT methods, the reparameterization-based method, low-rank adaptation (LoRA) (Hu et al., 2021), is considered one of the most effective methods for LLMs (Xu et al., 2023; Ding et al., 2022; Xin et al., 2024). Although LoRA and its more recent variants (Zhang et al., 2023a; Ding et al., 2023b; Hu et al., 2023) are effective and can bring stable downstream performance, they cause inference inefficiency under the multi-tenant setting (Chen et al., 2023), where one LLM backbone has to serve multiple users/tasks with the help of multiple sets of LoRA parameters and the LoRA parameters can not be merged to the LLM backbone. LoRA has to add low-rank modules to multiple linear weights of the Transformer layer, introducing significant additional latency in every token generation step. Thus, to promote efficiency in industrial usage, it is of vital importance to investigate the following research question:

***RQ1.*** *For a given LLM backbone, can we find a LoRA setting that adds as few fantastic LoRA modules as possible to ensure efficiency, and is this setting universally transferable to different industrial tasks?*

To address the above ***RQ1***, we now propose the Fantastic LoRA (FanLoRA) framework (Figure 2), which makes LoRA more suitable for industrial applications. First, we add LoRA modules with an equal rank to each Transformer weight (full LoRA setting) and fine-tune them on a general-purpose large-scale instruction tuning dataset ($D_{train}$). Second, after fine-tuning, we calculate the importance of each LoRA module via AB-score, a novel im-

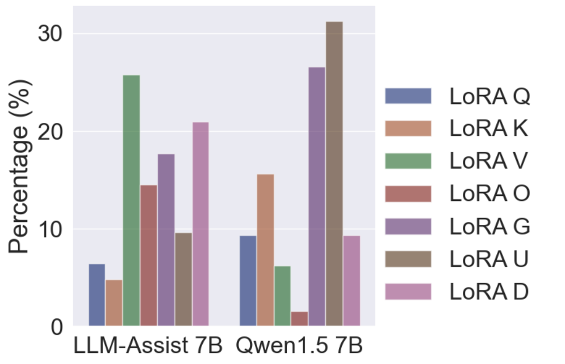


Figure 1: Distribution of LoRA modules across Transformer layers under the FanLoRA setting. The LLM backbones are LLM-Assist 7B and Qwen1.5 7B.

[*]Equal contributions.
[†] Corresponding author. Email: michaelwzhu91@gmail.com.

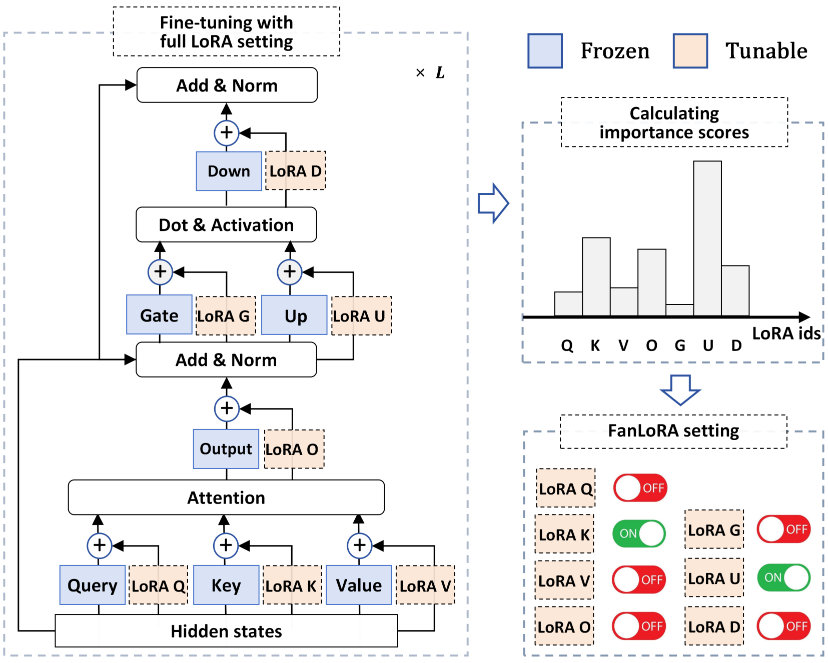


Figure 2: Schematic illustration of our FanLoRA framework.

portance scoring method we propose. Third, we keep at most two LoRA modules for each Transformer layer based on the importance scores and obtain the FanLoRA setting. In the fourth step, the LoRA modules under the FanLoRA setting are randomly initialized and fine-tuned on downstream tasks. Figure 1 presents the distribution of selected LoRA modules in the FanLoRA setting when the LLM backbone is Qwen1.5 7B[1] or LLM-Assist 7B, a proprietary LLM developed by an industrial participant[2].

We conduct extensive experiments on various open benchmark and proprietary tasks, including question answering, content generation, math reasoning, and general LLM evaluation, demonstrating that our FanLoRA setting can be widely applied to different downstream tasks. Our method can consistently outperform strong PEFT baselines with comparable tunable parameter budgets, especially the recent LoRA variants. Through our experiments and analysis, we can obtain the following takeaways: (a) FanLoRA demonstrates that one can effectively fine-tune the LLMs by adding a small number of LoRA modules. (b) The FanLoRA setting performs well on a wide range of downstream tasks for a given LLM backbone, demonstrating its broad transferability. (c) Our FanLoRA method has significantly lower latency than the previous LoRA variants, showing potential for wide industrial applications. Our contributions are summarized as follows:

- we propose a novel framework, FanLoRA, to evaluate each LoRA module via a novel importance scoring method and provide an efficient LoRA setting.
- We have conducted extensive experiments and analysis showing that our FanLoRA setting is effective and efficient under the multi-tenant setting and suitable for industrial usage.

## 2 Related works

LoRA (Hu et al., 2021) is proven to be an effective PEFT method when applied to both relatively small pretrained backbones and large language models (Dettmers et al., 2023; Zhu et al., 2023). Recently, many LoRA variants have been proposed. AdaLoRA (Zhang et al., 2023a) expresses the low-rank matrix multiplication of LoRA in the form of singular value decomposition (SVD), and it identifies the most critical ranks by a sensitivity-based importance score. SoRA (Ding et al., 2023b) prunes abundant LoRA ranks by imposing a $l_0$ norm and optimizing with proximal gradient descent. SaLoRA (Hu et al., 2023) prunes the LoRA ranks via the Lagrange multiplier method. Despite its

[1]`https://huggingface.co/Qwen/Qwen1.5-7B`

[2]The LLM-Assist 7B model has the same Transformer architecture with LlaMA-2 7B, but has a large vocabulary for supporting languages other than English. Due to policies on anonymous reviews, the detailed information for the LLM-Assist 7B model and its developers will be revealed upon acceptance.

tractability and effectiveness, LoRA still has room for improvement in both downstream task performances and efficiency under the multi-tenant setting (Chen et al., 2023).

Despite these recent efforts, the above LoRA variants still add LoRA modules to almost all the weight matrices in the Transformer backbone, which results in significant latency under the multi-tenant setting and poses difficulties for industrial usage. Our work complements the existing literature by addressing LoRA's efficiency issue.

## 3 Methods

### 3.1 Preliminaries

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

### 3.2 The FanLoRA framework

Now, we are ready to elaborate on the workflow of the FanLoRA framework.

**Full LoRA fine-tuning** As depicted in Figure 2, for each Transformer module $m$ in {Q, K, V, O, G, U, D} at layer $l$ ($l < L$, $L$ is the number of layers in the LLM), we add a LoRA module $m$ with rank size $r_0 > 0$ to reparameterize it. Formally, the forward calculation of module $m$ with LoRA is:

$$x^{'} = xW_{m,l} + xW^{A}_{m,l}W^{B}_{m,l} + b_{m,l}, \qquad (1)$$

where $W_{m,l} \in \mathbf{R}^{d_1 \times d_2}$ is the weight matrix of module $m$, $b_{m,l} \in \mathbf{R}^{1 \times d_2}$ is its bias term. $W^{A}_{m,l} \in \mathbb{R}^{d_1 \times r_0}$ and $W^{B}_{m,l} \in \mathbb{R}^{r_0 \times d_2}$ are the low-rank matrices for the LoRA module. In this step, we conduct LoRA fine-tuning on a general-purpose large-scale instruction fine-tuning datasets $D_{train}$ like UltraChat (Ding et al., 2023a), and the LoRA parameters will acquire knowledge of diverse tasks after fine-tuning.

**Evaluating the importance score of each LoRA module** In this part, we evaluate the importance of each LoRA module. A series of attribution methods are available, but they are not satisfactory for industrial usage. As pointed out by Held and Yang (2022), the sensitivity-based importance estimation by Michel et al. (2019) cannot distinguish whether it can improve or degrade the model so that pruning may be guided in the wrong direction in practical applications. The Shapley Value is widely applied in model interpretability (Zhao et al., 2024; Saha et al., 2022), primarily due to its sound theoretical foundation and properties (Lundberg and Lee, 2017). However, for large models like LLMs, the calculation of Shapley Value is intractable due to its computation complexity. To efficiently compute the importance score, we propose a novel method called ablation-based score (AB-score) since our method mimics conducting ablation studies for a LoRA module.

we introduce a binary LoRA gate $g_{m,l} \in \{0, 1\}$ into Equation 1:

$$x^{'} = xW_{m,l} + g_{m,l} * xW^{A}_{m,l}W^{B}_{m,l} + b_{m,l}, \qquad (2)$$

In the previous full LoRA fine-tuning step, all $g_{m,l}$ are set to 1. To compute the importance score of a given LoRA $m$ at layer $l$, We now consider four model settings with LoRA adaptations:

- $M_{all}$, which is exactly the model obtained from the previous step.
- $M_{\backslash(m,l)}$, which is obtained by only zeroing out the LoRA gate $g_{m,l}$ in $M_{all}$.
- $M_{null}$, where all LoRA gates are set to zero. That is, no LoRAs are added to the LLM.
- $M_{(m,l)}$, which is obtained by only setting the LoRA gate $g_{m,l}$ to 1 in $M_{null}$.

Denote the performance of model $M$ on the validation set $D_{val}$ as $S_{val}(M)$, then the importance score $V_{m,l}$ of LoRA $m$ at layer $l$ is given by

$$\begin{aligned} V_{m,l} =& S_{val}(M_{all}) - S_{val}(M_{\backslash(m,l)}) \\ &+ S_{val}(M_{(m,l)}) - S_{val}(M_{null}). \end{aligned} \qquad (3)$$

Note that for a given LLM, $S_{val}(M_{null})$ and $S_{val}(M_{all})$ are fixed, so the above equation can be simplified as $V_{m,l} = -S_{val}(M_{\backslash(m,l)}) + S_{val}(M_{(m,l)})$. We will use experiments to demonstrate that our method AB-score is comparable to the Shapley value and better than the sensitivity-based method.

**Obtaining the FanLoRA setting** After obtaining the importance score for each LoRA module, we perform pruning on the LoRA modules of each Transformer layer with the following principles:

- We keep at most top $K_{max}$ LoRA modules ($K_{max} > 0$) with the highest scores in each Transformer layer.
- If the importance score $V_{m,l}$ of a LoRA module is negative, it will be pruned.[3]

We will refer to the LoRA setting obtained from the above steps as the FanLoRA setting. In the next section, we will use experiments to show that, for a given LLM backbone, the FanLoRA setting is applicable for a wide range of tasks.

**Adaptation for downstream Tasks** For various downstream tasks, we fine-tune each task using the same FanLoRA setting obtained from the previous steps. The LoRA modules with rank size $r_1 > 0$ are added to the Transformer backbone according to the FanLoRA setting, and their parameters are randomly initialized and fine-tuned on the given task. We will evaluate the effectiveness, efficiency, and universality of the FanLoRA setting through the performance of various downstream tasks.

## 4 Experiments

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

### 4.1 Baselines

We compare our FanLoRA framework with the current SOTA PEFT baseline methods: (a) (IA)$^3$ (Liu et al., 2022), 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. The hyper-parameter settings for the baselines are detailed in Appendix C.

### 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 infrastructure** We run all our experiments on NVIDIA A40 (48GB) GPUs.

**Pretrained backbones** The main experiments use a proprietary LLM, LLM-Assist 7B, as the pretrained backbone model. We also run the FanLoRA framework with Qwen1.5 7B[4].

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

**Settings for the FanLoRA framework** In this work, for the full LoRA fine-tuning step of FanLoRA framework, we add LoRA modules with rank $r_0 = 12$ at each linear module of the Transformer block. The large-scale UltraChat (Ding et al., 2023a) dataset is split into a train set $D_{train}$ and a development set $D_{val}$, with a ratio of 99:1. $D_{train}$ is used to fine-tune the LoRA modules, and $D_{val}$ is used to calculate the importance scores. For the downstream adaptation step of FanLoRA, each Transformer block keeps at most $K_{max} = 2$ LoRA module, and the rank of LoRA modules is set to $r_1 = 12$. Under the above settings, our FanLoRA method will introduce 8.8M tunable parameters to the LLM-Assist 7B backbone.

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

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

### 4.4 Main results

The experimental results on the SQuAD, BoolQ, COPA, HSM10K, and Q2SQL tasks are presented in Table 1, in which the number of tunable parameters is reported in the second column. Table 1 reveals that our FanLoRA method outperforms the baseline methods across all five tasks, with comparable or fewer tunable parameters. In particular,

[3]According to this principle, the number of the kept LoRA modules in a Transformer layer may be smaller than $K_{max}$.

[4]`https://huggingface.co/Qwen/Qwen1.5-7B`

| 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*** | | | | | | |
| FanLoRA | 8.6M | **56.4** | **83.1** | **88.9** | **87.9** | **92.4** |

Table 1: 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.

| Method | MT-Bench gpt4-score (↑) | MMLU acc | LLM-Eval1 acc |
|---|---|---|---|
| AdaLoRA | 7.13 | 46.5 | 56.8 |
| FanLoRA | 7.28 | 47.9 | 58.9 |

Table 2: Performance of general-purpose instruction tuning using the FanLoRA and AdaLoRA methods. The backbone model is LLM-Assist 7B. ↑ means the metric is higher the better.

FanLoRA outperforms previous SOTA LoRA-style baselines, LoRA and AdaLoRA, with comparable parameters.

After the LLM-Assist 7B is fine-tuned on the UltraChat (Ding et al., 2023a) dataset with our FanLoRA setting or the AdaLoRA methods, we utilize the challenging benchmarks, MT-Bench, MMLU, and LLM-Eval1, for evaluation. The experiments are conducted under the zero-shot setting, and no demonstrative examples are concatenated to the prompts. Table 2 presents the results. Consistent with the previous experiments (Table 1), our FanLoRA method outperforms the AdaLoRA methods on the three benchmarks, demonstrating that FanLoRA is superior in enhancing the instruction tuning quality of large language models.

The above results demonstrate that our FanLoRA framework has successfully addressed ***RQ1*** in Section 1: FanLoRA adds only two LoRA modules per Transformer layer to reduce inference latency and performs well on a wide range of downstream tasks.

## 4.5 Ablation studies and analysis

**Visualization and analysis of the FanLoRA setting** Figure 1 presents the proportion of each LoRA module across the Transformer layers under the FanLoRA setting when the LLM backbone is LLM-Assist 7B or Qwen1.5 7B. We also present the corresponding detailed LoRA importance scores and FanLoRA setting as heatmaps in Figure 5 and 6 in Appendix D. We can observe that: (a) In the FanLoRA setting, the distribution of LoRA modules across the Transformer layers is unbalanced. In LLM-Assist 7B, six layers choose to add LoRA modules on the Query or Key module, while 16 layers select the Value module to add LoRA. (b) The LoRA importance distributions at different layers differ inside a given LLM backbone. Intuitively, different Transformer layers play different roles, and their knowledge is expressed in different linear modules, causing LoRA modules to have different importance. (c) Although fine-tuned on the same dataset, the LoRA importance distributions on the Qwen1.5 7B model differ from those on LLM-Assist 7B. However, a few common characteristics can be observed: on the lower layers, the LoRA modules in the self-attention part receive higher importance scores, while on the deeper layers, the FFN part's LoRA modules are generally more important.

**Analysis of the inference efficiency** To demonstrate the inference efficiency of our FanLoRA method, we now compare the GPU memory and generation speed of FanLoRA, AdaLoRA, and $(IA)^3$. In this experiment, LoRA parameters are not merged to the backbone to mimic the single-LLM multi-tenant setting (Chen et al., 2023) in industry applications. The detailed settings for efficiency analysis are presented in Appendix B. From Table 3, one can see that: (a) Our FanLoRA method and $(IA)^3$ have comparable tunable parameters, memory costs, and generation speed during generation.

| Method | Beam size | Speed (tps) | Memory cost (MiB) |
|---|---|---|---|
| $(IA)^3$ | 1 | 33.1 | 14572 |
| | 3 | 27.6 | 16036 |
| AdaLoRA | 1 | 25.1 | 14616 |
| | 3 | 21.9 | 16104 |
| FanLoRA | 1 | 31.8 | 14576 |
| | 3 | 26.7 | 16054 |

Table 3: The memory and speed of LLM-Assist 7B for generating responses with different PEFT methods.

| | Seed 1 | Seed 2 | Seed 3 |
|---|---|---|---|
| Seed 1 | 1.00 | 0.989 | 0.984 |
| Seed 2 | - | 1.00 | 0.991 |
| Seed 3 | - | - | 1.00 |

Table 4: The pairwise correlation scores for the LoRA importance estimations obtained under three random seeds.

(b) Our FanLoRA is much faster than AdaLoRA. The LoRA-based method requires the model to call the LoRA modules at each token generation step. Since FanLoRA has significantly fewer LoRA modules than the AdaLoRA method, its inference speed will be superior.

**On the stability of FanLoRA setting** On a given LLM backbone, we must investigate whether the FanLoRA setting is stable under different random seeds. We run the FanLoRA framework under three different random seeds and then calculate the similarity of the importance scores, measured using Spearman rank correlation. Note that these three results are not included in the previous experiments. Table 4 presents the pairwise similarity scores. The results show that the importance scores of the LoRA modules obtained under different random seeds have very high correlations, indicating that the FanLoRA setting obtained by our FanLoRA method is stable against random seeds.

**Effects of $K_{max}$** In Table 1, we set the number of kept LoRAs per layer, $K_{max}$, to 2, in order to achieve higher efficiency. Now, we alter $K_{max}$ to {1, 3, 4, 5, 6, 7}. The rank parameter $r_1$ is adjusted accordingly, from 12 to {24, 8, 6, 5, 4, 4}, so each setting has a comparable number of tunable parameters. The results of the BoolQ and Q2SQL tasks are presented in Figures 3(a) and 3(b). The results show that: (a) The best performance occurs with $K_{max} = 2$ for both tasks, validating our default experimental setting (in Table 1). (b) With the increased number of kept LoRA modules, FanLoRA's performance first increases and then decreases. When $K_{max}$ reaches 7, FanLoRA reduces to the vanilla LoRA. The results are intuitive.

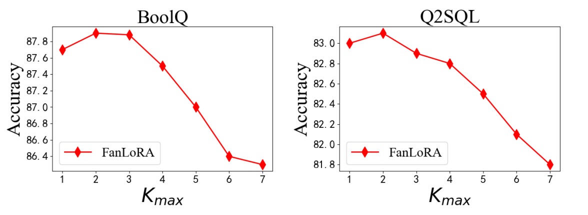


Figure 3: Performances under different values of $K_{max}$, the maximum number of LoRA modules kept per layer.

When $K_{max}$ increases from 1 to 2 or 3, we include LoRA modules that are the most effective, enhancing the fine-tuning performance. However, when $K_{max}$ keeps increasing, many LoRA modules with negative impacts are included and will degrade the downstream performance.

**Ablation on the FanLoRA framework** Table 6 of Appendix E demonstrate that: (a) Compared to our AB-score method, the sensitivity-based method (Michel et al., 2019) is less effective in identifying the most important LoRA modules that need to be kept, resulting in less effective LoRA settings. (b) a large-scale instruction tuning dataset like UltraChat is essential for our FanLoRA framework to perform well. And the small-scale instruction tuning dataset like Alpaca (Taori et al., 2023) is not enough.

**More ablation studies** (a) Figure 4 of Appendix F demonstrate that the FanLoRA method consistently outperforms AdaLoRA under different budgets of tunable parameters. (b) Table 7 in Appendix F demonstrates that our FanLoRA framework works well with different LLM backbones.

## 5 Conclusion

In this work, we introduced the Fantastic LoRA (FanLoRA) framework to enhance the efficiency of parameter-efficient fine-tuning (PEFT) for large language models (LLMs) in industrial applications. By using a novel AB-score method to identify the most critical LoRA modules, FanLoRA effectively reduces latency overhead while maintaining high performance across diverse downstream tasks. Our extensive experiments demonstrate that FanLoRA outperforms existing PEFT baselines with comparable tunable parameters, proving its versatility and efficiency in multi-tenant settings where an LLM backbone has to serve multiple users/tasks via different sets of LoRA parameters.

## Limitations

We showed that our proposed method can improve the performance and efficiency of parameter-efficient tuning on diverse tasks and different LLMs, thus can help to reduce the cost of industrial applications involving LLMs. However, we acknowledge the following limitations: (a) the more super-sized open-sourced LLMs, model with 20B or 70B parameters, are not experimented due to limited computation resources. (b) Other tasks in natural language processing, like information extraction, were also not considered. But our framework can be easily transferred to other backbone architectures and different types of tasks. It would be of interest to investigate if the superiority of our method holds for other large-scaled backbone models and other types of tasks. And we will explore it in future work.

## Ethics Statement

The finding and proposed method aims to improve the LoRA based tuning in terms of better downstream performances whiling pursuing efficiency. We make sure that the used datasets are fully anonymized and have gone through thorough ethical checks. In this work, we have experimented with both open-sourced and proprietaty LLMs. As with all LLMs, These models' 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 investigate different fine-tuning methods for these LLMs, not building applications directly using these models. In the future, we would like to conduct further tests to see how our method affects the safety aspects of LLMs.

## References

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 Fine-tuning of Quantized LLMs. *arXiv e-prints*, page arXiv:2305.14314.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023a. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. 2023b. Sparse low-rank adaptation of pre-trained language models. In *Conference on Empirical Methods in Natural Language Processing*.

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.

William Held and Diyi Yang. 2022. Shapley head pruning: Identifying and removing interference in multilingual transformers. *arXiv preprint arXiv:2210.05709*.

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

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

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

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

Yahao Hu, Yifei Xie, Tianfeng Wang, Man Chen, and Zhisong Pan. 2023. Structure-aware low-rank adaptation for parameter-efficient fine-tuning. *Mathematics*.

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

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

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.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *NeurIPS*.

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

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.

Sourav Saha, Debapriyo Majumdar, and Mandar Mitra. 2022. Explainability of text processing and retrieval methods: A critical survey. *arXiv preprint arXiv:2212.07126*.

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, 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.

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.

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

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

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*.

Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. 2024. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*, 15(2):1–38.

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.

| Datasets | #train | #dev | #test | $\vert\mathcal{Y}\vert$ | 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 5: The dataset statistics of the GLUE and SuperGLUE benchmark tasks evaluated in this work. $|\mathcal{Y}|$ is the number of classes for a classification task.

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

## A 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 5.

**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 5.

**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., 2023a) 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.

## B Appendix: settings for efficiency analysis

In the Table 3 of the main contents, we conduct analysis on the FanLoRA and other PEFT methods' memory and speed during inference. We present two metrics for measuring efficiency: (a) peak memory cost during generation. (b) tokens generated per second (tps).

We restrict the number of newly generated tokens to be 32 under the method of beam search with beam size equal to 1 or 3. The length of the initial instruction is 276 under the tokenizer of the LLM-Assist 7B model. We run the generation process for 100 times to calculate the average metric values, reducing the randomness.

## C Appendix for Experimental settings

Here, we provide more details for experimental settings.

**Hyper-parameters for the baseline PEFT methods** For the P-tuning method, the soft prompts' length is 64, and the soft prompts is first initialized with dimension 36, and then a learnable projection layer projects it to the same dimension with the LLM-Assist 7B backbone. For P-tuning V2, the number of prompt tokens at each layer is set to 64. For LPT and IDPG, the bottleneck dimension is set to 1024, and the number of soft tokens is set to 4.

For the Houlsby-Adapter, the bottleneck dimension is set to 18, and the adapter modules are added on the self-attention and feed-forward module. For the Learned-Adapter, the bottleneck dimension is set to 36, and the adapter modules are connected to the whole block.

We adjust the sparsity for SSP so that the number of tunable parameters is comparable with FanLoRA and the other baselines.

For $(IA)^3$, the activation adjusting vectors are added the Query, Key, and Up activations. The adjusting vectors are initialized with dimension 16, and then a learnable projection layer projects it to the same dimension with the LLM-Assist 7B backbone.

For LoRA, the initial rank at each module is set to 4. For AdaLoRA, the initial rank at each module is set to 8, and half of the rank budget is pruned during fine-tuning.

**Training settings for PEFT methods** We use the HugginFace Transformers (Wolf et al., 2020b), PEFT (Mangrulkar et al., 2022), or the original code repositories for implementing all the methods, and for training and making predictions. For fine-tuning LLM-Assist 7B model, the maximum sequence length is set to 1024. The maximum training epoch is set to 10 on the downstream tasks. For fine-tuning on UltraChat, the training epoch is set to 1. The batch size is set between 16 for task with less than 10k training set, and 128 otherwise. We use AdamW as the optimizer with a linear learning rate decay schedule and 6% of the training steps for warm-up. The learning rate is set to 1e-4. The other hyper-parameters are kept the same with Wolf et al. (2020b). In every 200 steps, the model is evaluated on the dev set to calculate dev set perplexity. Patience is set to 10, that is, if the model does not achieve a lower dev set perplexity for 10 evaluation runs, the training stops early. The best checkpoint on the dev set is used to run predictions on the test set.

## D Visualization of the FanLoRA settings

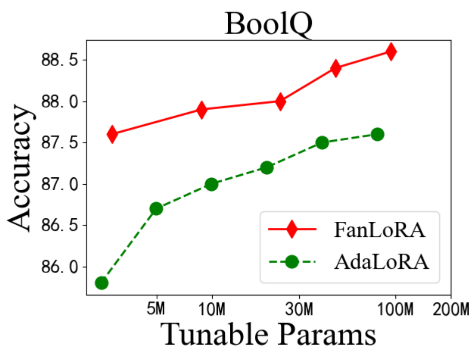
In Figure 5, we present LoRA importance scores on LLM-Assist 7B and Qwen1.5 7B. In Figure 6, we present the FanLoRA setting on LLM-Assist 7B and Qwen1.5 7B.

## E Ablation on the FanLoRA framework

We now consider the following variants of the FanLoRA framework: (a) substituting the large scale instruction tuning dataset $D_{train}$ from UltraChat to Alpaca (Taori et al., 2023). The latter is two orders of magnitude smaller than the former. We denote this version as FanLoRA-1. (b) FanLoRA-2, which uses the sensitivity based importance scoring method (Michel et al., 2019) instead of the AB-score. The experiments on the BoolQ and Q2SQL tasks are presented in Table 6. The results show that FanLoRA under the default settings (as in Table 1) outperforms the two variants. In addition: (a) Comparing FanLoRA to FanLoRA-1 demonstrates that a large scale instruction tuning dataset is essential for our FanLoRA framework to perform well. (b) Comparing FanLoRA to FanLoRA-2 shows that the sensitivity based method (Michel et al., 2019) is less effective in identifying the most important LoRA modules that need to be kept.

| Method | BoolQ (acc) | Q2SQL (acc) |
|---|---|---|
| FanLoRA | **87.9** | **83.1** |
| FanLoRA-1 | 86.2 | 82.6 |
| FanLoRA-2 | 87.1 | 42.3 |

Table 6: The comparison of FanLoRA's variants on the BoolQ and Q2SQL tasks. The backbone model is LLM-Assist 7B.



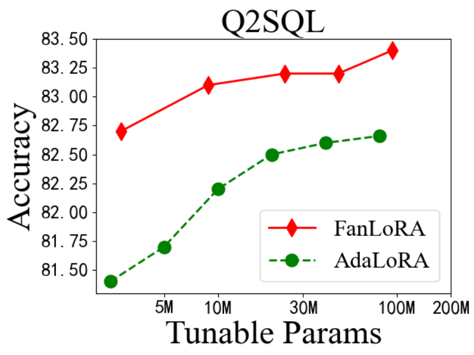


Figure 4: Performances under different tunable parameter budgets. The $x$-axis represents the number of tunable parameters, and the $y$-axis represents the performance score.

## F More ablation studies

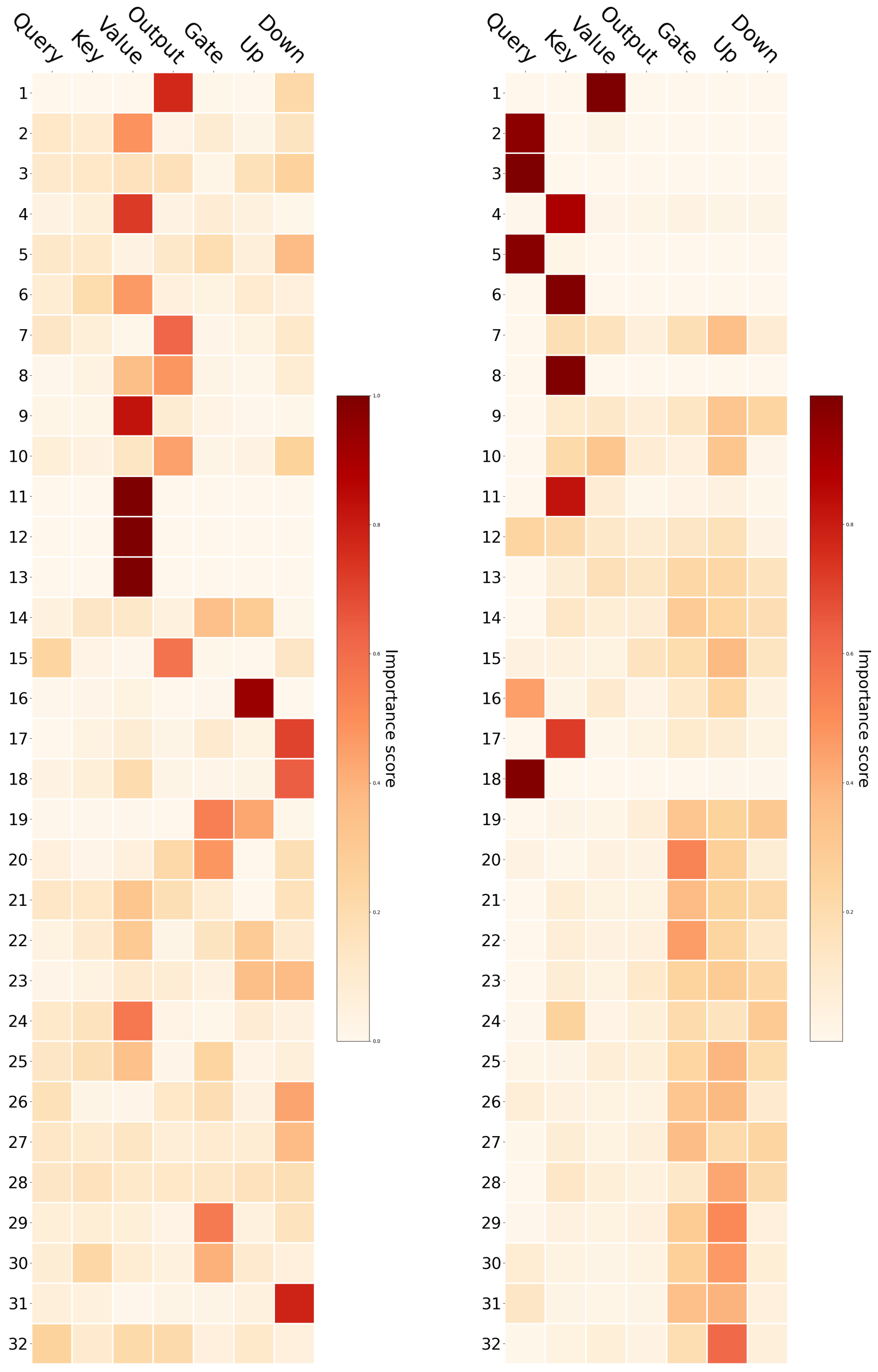
**Comparisons under different budgets of tunable parameters** We vary the budget of tunable parameters for FanLoRA by modifying the LoRA rank value of $r_1 = 12$ to {4, 32, 64, 128}. We also vary the AdaLoRA method' tunable parameter numbers. The experimental results on the BoolQ and Q2SQL tasks are presented in Figure 4(a) and 4(b). The results show that under different tunable parameter budgets, our FanLoRA method can consistently outperform the AdaLoRA method.

**Ablation on the LLM backbones** Our main experiments (Table 1) are conducted on the LLM-Assist 7B model. To demonstrate the broad applicability of our method, we now conduct experiments

| **Method** | **BoolQ (acc)** | **Q2SQL (acc)** |
|---|---|---|
| ***Results for LlaMA-2 7B*** | | |
| AdaLoRA | 84.9 | 80.8 |
| FanLoRA | **86.4** | **81.7** |
| ***Results for Qwen1.5 7B*** | | |
| AdaLoRA | 85.7 | 81.5 |
| FanLoRA | **87.1** | **82.6** |

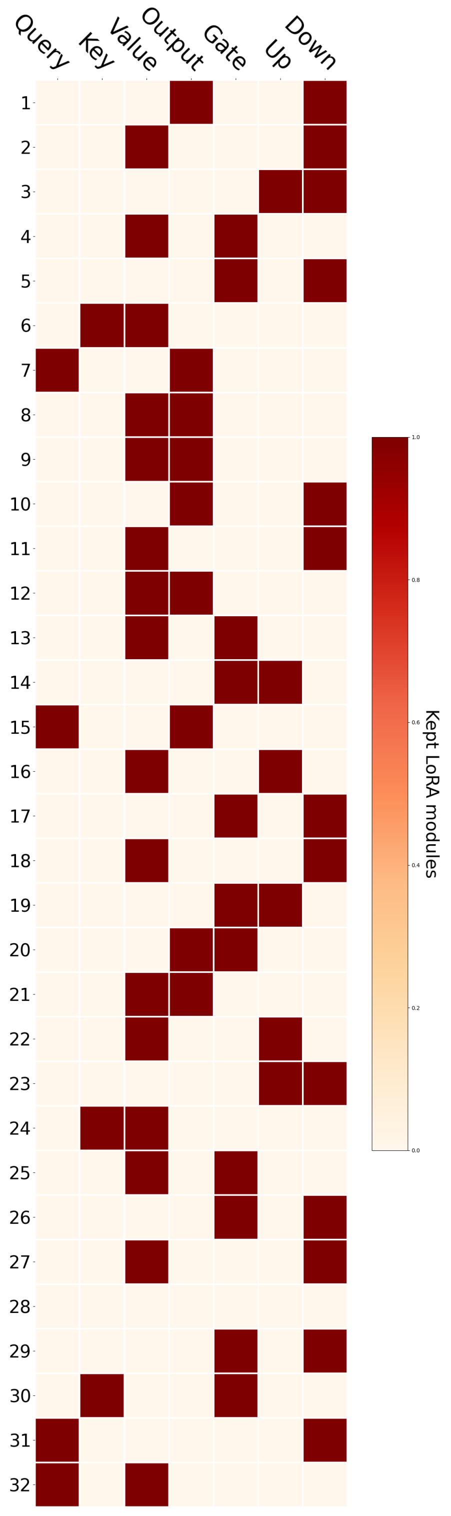
Table 7: Results for different PEFT methods on the BoolQ and Q2SQL benchmarks. The backbone LLMs are LlaMA-2 7B and Qwen1.5 7B.

on LlaMA-2 7B and Qwen1.5 7B. The results are reported in Table 7. We can see that on these three backbones, our FanLoRA method can also outperform the baseline PEFT methods.

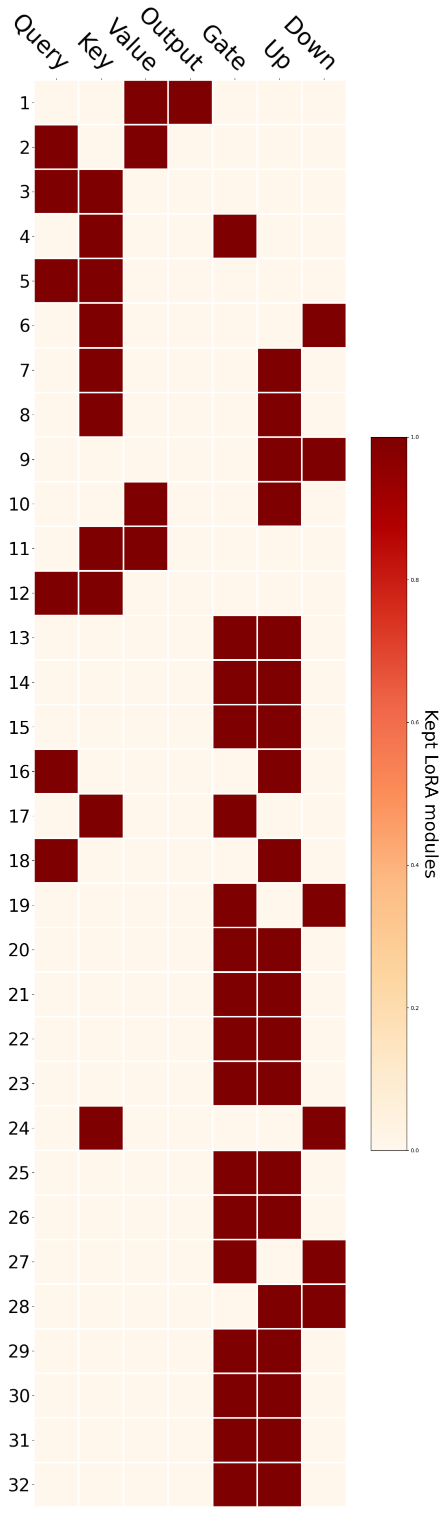


(a) LLM-Assist 7B (b) Qwen1.5 7B

Figure 5: The LoRA importance scores on LLM-Assist 7B and Qwen1.5 7B.



(a) LLM-Assist 7B



(b) Qwen1.5 7B

Figure 6: The FanLoRA settings on LLM-Assist 7B and Qwen1.5 7B.