Exploring The Principles and Prospects for Efficient Fine-Tuning of Transformer-Based Pre-Trained Large Language Models

Ruiqi He

FOB, City University of Macau, Macau, China

Abstract. In recent years, large language models (LLMs) have made breakthroughs in natural language processing and multimodal tasks. However, the growing model size and the high cost of full parameter fine-tuning pose challenges to their efficient adaptation. This paper focus on Transformer-based Parameter Efficient Fine-Tuning (PEFT) techniques for large models, and analyze three types of methods, namely, additive-based, specification-based and reparameterization-based, from the perspectives of performance, engineering complexity and applicability. This paper concludes that the PEFT technique exhibits comparable or even better performance than full parameter fine-tuning in a variety of tasks, but still faces stability and adaptability challenges in complex scenarios. Future research can further advance the field by improving flexibility, optimizing strategies and focusing on privacy and security. The purpose of this paper is to provide a basic reference for researchers to understand the PEFT algorithm and its system implementation, to further promote the implementation of PEFT technology in the research industry.

1 Introduction

In recent years, Large Language Models (LLMs) have made significant progress in the field of Natural Language Processing (NLP), and are widely used in tasks such as text generation, machine translation , dialog systems, and erasure generation [1-4].LLMs have pushed forward the development of Artificial Intelligence (AI) by demonstrating near-human level text comprehension and generation through the linguistic patterns of neural networks. However, with the increasing scale of the models, their high computational cost and storage requirements make efficient fine-tuning (fine-tuning) an important challenge. How to reduce the fine-tuning cost while maintaining the model performance has become a hot issue in current research.

The traditional full fine-tuning method requires updating all the parameters of the model, which has a huge computational overhead for large-scale LLMs and is difficult to be efficiently deployed in resource-constrained environments. To solve this problem, researchers proposed the Parameter- Efficient Fine-Tuning (PEFT) technique, which adjusts only a small portion of the model's parameters while keeping most of the parameters

b23090115370@cityu.edu.mo

unchanged, thus dramatically reducing the computational overhead and the storage requirements and enabling the LLMs to be adapted to new tasks more efficiently [5]. In addition, the applicability of PEFT techniques has extended beyond the NLP domain, with success in computer vision (CV) tasks such as Vision Transformers (ViTs) [6] and Vision-Language Models (VLMs) [7].

This paper systematically reviewed and classified the latest research progress on PEFT, summarized and analyzed the methods from the perspectives of computational efficiency, adaptation strategies and practical application scenarios. Section 2 introduces the basic concepts of Large Language Model (LLM) pretraining, including the core Transformer architecture and its latest research results. Section 3 focuses on the mainstream PEFT methods and classifies them into additive, specification, and reparameterization method, according to the way of architecture adjustment [8]. Section 4 analyzes the global performance of various PEFT techniques in different scenarios, considering the existing experiments and research results. Section 5 summarizes the core contents of the review and looks forward to the future development direction. The purpose of the paper is to integrate the current technical achievements and comprehensively analyze the advantages and disadvantages of the existing methods, in the hope of providing references for optimizing the adaptation strategies of large-scale models and advancing the related research.

2 Transformer

Transformer [9] has become the de facto base module for pre-training large models today. The original Transformer contains two modules: an encoder and a decoder. Both use the attention mechanism as the core method for capturing key contextual information in a sequence[10,11]. When processing sequences, traditional Recurrent Neural Networks (RNN) or Convolutional Neural Networks (CNN) often struggle to capture the importance of different locations in the input sequence. In contrast, an attentional mechanism can interact with all other locations based on the hidden state of any location in the sequence, thereby assigning higher weights to the more important parts and ignoring the less important parts.

The Transformer's Self-Attention is in the form of Scaled Dot-Product Attention, given a query matrix Q, a key matrix K, and a value matrix V. Each row corresponds to a position in the sequence Query/Key/Value. Given a query matrix Q, a key matrix K, and a value matrix V, each row corresponds to a query/key/value at a certain position in the sequence, and the self-attention is computed as follows：

$$\mathrm{ATT}(Q,K,V) = \mathrm{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

Where， QK^T characterizes the degree of matching between any two positions in the sequence.

To allow the model to capture more fine-grained dependencies from different 'attention subspaces', Transformer introduces the Multi-Head Attention (MHA) mechanism. There will be different linear projection for X：

$$Q = XW_q, \quad K = XW_k, \quad V = XW_v \tag{2}$$

Where, Wq , Wk , Wv are the trainable parameters of the I th head。For each head, there will be a set of attention outputs Hi. Then concat each head’s output and multiplied by Wo to project back to original space and get the final output.

$$H = \text{MH-ATT}(Q, K, V) = \text{Concat}(H_1, \ldots, H_h)W_o \quad (3)$$

$$H_i = \text{ATT}\left(Q_{(i)}, K_{(i)}, V_{(i)}\right) = \text{ATT}\left(XW_{q_{(i)}}, XW_{k_{(i)}}, XW_{v_{(i)}}\right) \quad (4)$$

Multi-head attention enhances the expressive power of the model by capturing the dependencies of different aspects of the sequence through multiple subspaces thereby.

3 DELTA FINE-TUNING

The purpose of fine-tuning is that for a given set of pre-trained model parameters and training dataset, the model needs to achieve a better performance in the downstream tasks on this dataset by learning to update the parameters and then is now on that dataset. However, traditional global fine-tuning techniques make this training process costly and difficult. Delta fine-tuning achieves downstream task performance comparable to global fine-tuning by training a small number of parameters through operations such as freezing most of the parameters or adding new ones [8]. This technique makes it less difficult to fine-tune large models, allowing more researchers to experiment and apply large models to more specialized downstream tasks. Delta fine-tuning techniques can be classified into four categories in terms of their implementations - Additive, which is achieved by adding new modules that can be trained; Selective, which is achieved by freezing most of the model's parameters and allowing a small portion of the parameters to be fine-tuned; and Selective, which is achieved by freezing most of the model's parameters and allowing a small portion of the parameters to be fine-tuned for training. Selective methods, which are realized by freezing most of the model parameters and allowing a small number of parameters to be fine-tuned for training; Reparametrized methods, which are realized by low-rank decomposition of the parameter matrix; and Hybrid methods, which are a mixture of the above three methods. In this section, typical models in these four categories are described in detail to show their technical paths and differences.

3.1 Additive-based

There are two most common ideas for efficient fine-tuning techniques based on the Additive idea: one is by adding an adapter module to the model; the other is the prompt method by adding a soft prompt to the input layer.

3.1.1 Adapter

Adapter [12], as a plug-in lightweight network structure, is widely used in large model fine-tuning. Compared to vanilla finetuning, adapter requires only a small number of parameters to be adjusted to accomplish the adaptation for downstream tasks, thus drastically reducing the cost of model training and deployment. In Neil Housley et al., the traditional adapter module can achieve similar or even better results than parameter fine-tuning by adding small-scale network layers in the shape of 'bottlenecks' in each transformer block and updating the parameters inside the adapter through the backpropagation process. The architecture of the Adapter module is illustrated in the figure: the Adapter module has the following components:

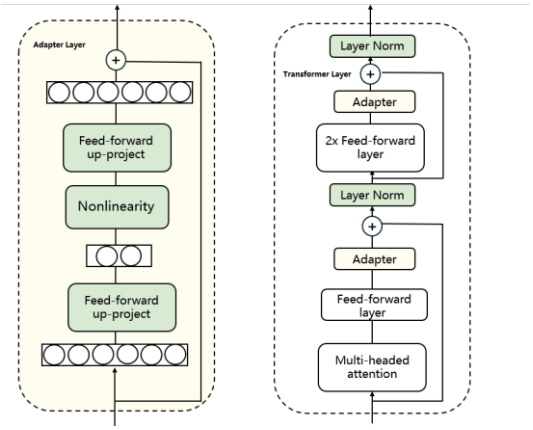

Fig. 1 Schematic diagram of the structure of the Adapter module with its insertion position in the Transformer network(Photo credit : Original).

In Figure 1, the left side shows the specific composition inside the Adapter, including down-projection, nonlinear activation, and up-projection; the right side shows how the Adapter is integrated in the Transformer layer, inserted between the multi-head attention mechanism and the feed-forward sublayer through residual connections [12] .

First, in the down-projection phase, the input feature vector h is projected into a low-dimensional space by a linear transformation W_{down} , where r denotes the bottleneck dimension of this Adapter module. Subsequently, a nonlinear activation function is attached to introduce the nonlinear transformation. Next, in the up-projection stage, the low-dimensional representation after the down-projection process is mapped back to the original feature dimensions by the linear transformation W_{up} to obtain an output vector with the same dimensions as the input features.

After that, the residual connection is appropriately applied to add the input directly to the output of the projection, which can be expressed in the following computational equation:

$$\text{Adapter}(x) = W_{\text{up}}\sigma(W_{\text{down}}x) + x \tag{5}$$

Such a residual design ensures that the information flow of the original backbone network is not disrupted when Adapter is added, while reinforcing the gradient transfer. Compared to the full parameter update, the Adapter module contains only a small number of projected layer parameters that can be trained. Similar ideas have given rise to many innovative structures and improved algorithms. For example, Ladder Side-Tuning (LST) [13] improves the performance of the original pre-trained model by using the adapter module as a side network. This approach avoids traversing the entire backbone network of the model in backpropagation and significantly reduces the memory footprint during training.

3.1.2 Prefix Tuning

Prefix tuning [14] keeps the parameters of the backbone network frozen by adding trainable prefixes to the multilayer hidden state of the Transformer and updating only these prefixes during gradient descent.

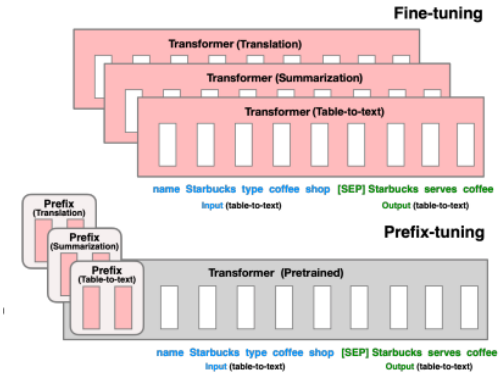

Fig. 2 Schematic diagram comparing the structure of Fine-tuning and Prefix-tuning in downstream task adaptation[14].

Fine-tuning needs to fine-tune the whole Transformer model for each task(Fig.2), while Prefix-tuning only adds a small number of learnable prefixes before the pre-trained model, which reduces the amount of parameter updating and improves multi-tasking adaptability [14].

The method first achieved good performance gains on autoregressive models such as T5, as well as seq2seq models such as BART. It is shown that under large-scale conditions, Prefix Tuning's performance on multiple downstream tasks can approach that of full parameter fine-tuning. However, the experiments also point out that Prefix Tuning is sensitive to the length of cue vectors as well as initialization, which usually requires a carefully designed initialization strategy, such as Kaiming Initialization, or the addition of extra hints (PPTs).

3.1.3 Prompt Tuning:

Prompt Tuning [15] adds trainable prompt vectors before and after the input, freezes the large model backbone, and updates only the prompt parameters, thus significantly reducing the parameter overhead. The principle of Prompt Tuning is similar to that of Prompt Tuning, but the main difference is that Prompt Tuning only adds trainable soft prompts to the input layer instead of inserting prefixes in the intermediate layer of the Transformer. The main difference is that Prompt tuning only adds trainable soft prompts in the input layer, instead of inserting prefixes in the intermediate layer of the Transformer.

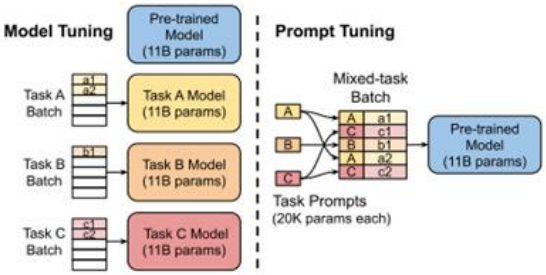

Fig. 3 Schematic diagram comparing the structure of Model Tuning and Prompt Tuning in multi-task adaptation[15].

As shown in Fig.3, model Tuning requires the complete model to be fine-tuned separately for each task, while Prompt Tuning achieves the sharing of the same pre-trained model across all tasks by introducing lightweight task prompts, which significantly reduces the parameter cost and improves the multitasking capabilities [15]

3.2 Specification-based

Unlike the method of inserting new modules into the model or adding trainable hints to the input, the specification-based approach does not introduce any new parameters but rather fine-tunes some of the existing parameters by specifying them directly inside the pre-trained model, leaving most of the remaining parameters frozen. This approach leaves the structure of the model intact, and selectively updates a small number of internal weights or biases of the original model during the training process to adapt to the needs of the downstream task. Since this approach drastically reduces the number of parameters that need to be updated while preserving the core expressive power of the backbone network, there are scenarios in which the use of specification can achieve performance like that of full-parameter fine-tuning.

Early specification-based approaches were based on heuristics [16] to artificially define which parameters are to be fine-tuned, and used the experience gained from previous experiments and continuous experimentation to specify which parameters are to be involved in the fine-tuning and which are to be frozen. The model can be fine-tuned using the experience gained from previous experiments and continuous trial-and-error to specify which parameters of the model are involved in fine-tuning and which ones are frozen. lee et al achieved around 99% full parameter fine-tuning performance by fine-tuning only the last two layers of the BERT/RoBERTa. BitFit [17] demonstrated that adjusting only the bias term (bias) can be used to bring the model all the way to completion by training the model to perform on a very small portion (0%) of the downstream task. A very small fraction (0.1%) of the parameters of the downstream task can be fine-tuned. These experimental results show that if the subset of parameters that need to be updated can be properly selected, the model can still achieve better performance on the downstream task even if most of the weights are dropped from the update. Observations from some studies (Functional Diversity) point out [18] that in specific implementations, even the same bias term may have different functionalities in different layers or modules. As a result, simple heuristic schemes often fail to find the optimal subset of parameters precisely, especially in larger scales or more complex networks, which introduces greater uncertainty in manual selection.

3.3 Reparameterization

The starting point of the Reparameterization-based approach is that during the fine-tuning of the model, most of the downstream task-induced weight changes satisfy the low-rank assumption [19], the parameter updates required for the fine-tuning can be done in a subspace much smaller than the original model dimensions, which drastically reduces the number of trainable parameters and the difficulty of optimization. It was found that the pre-training process implicitly compresses the 'intrinsic dimensionality' of the model; in the fine-tuning phase, if the changes in the weights of the model can be projected onto this low-dimensional space or low-rank structure, similar results to full parameter fine-tuning can be achieved by updating a small number of parameters.

Research showed experimentally that about (85%) fine-tuning performance can still be obtained when mapping the process of full parameter fine-tuning to a subspace whose values contain a thousand dimensions, which side-steps the fact that large-scale pre-trained models often need to be updated with a limited 'number of intrinsic parameters' when fine-tuning. Moreover, as the size of the model increases, its so-called intrinsic dimension may further become smaller, implying that large models possess stronger potential for parameter compression, and a variety of downstream tasks can be accomplished with fewer trainable parameters under low-rank assumptions.

LoRA [20] assumes that the change of the weights in transformer can be represented by a low rank matrix δW. Compared to traditional full parameter fine-tuning, LoRA only

additionally introduces the decomposition matrices A and B (of relatively low rank), such that:

$$\delta w = A * B \tag{6}$$

During the stage of forward propagation, this decomposition matrix is multiplied and added to the original weights when the self-attention layer performs a linear transformation of the input, i.e.:

$$W_{update} = W_{pretrained} + A * B \tag{7}$$

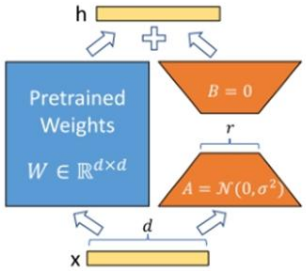

Fig. 4 Schematic diagram of the structure based on the introduction of low-rank perturbations with pre-training weights[20].

By fixing the pre-training weights and introducing a low-rank perturbation r consisting of a random initialization matrix A and a zero matrix B, the improved task adaptation capability of the model is achieved while keeping the parameters efficient [20].

Due to the smaller dimensions of A and B, the number of parameters that need to be updated to find the optimal solution is significantly reduced. Training and Inference Process(Fig.4).

1) Training phase: after freezing the self-attention weights of the original model, only the A and B matrices introduced by LoRA are updated. Since the rank r is much smaller than the original weight matrix, the training cost can be reduced significantly.

2) Inference phase: inference again only requires superposition of A and B at the self-attention weights, which has minimal impact on the original structure of the model and generally adds no additional inference overhead: AB can be pre-merged into $W_{pretrained}$.

Reparameterization-based method successfully balances parameter efficiency and performance by mapping the weight changes in the fine-tuning phase of the model to a low-rank or low-dimensional subspace. In particular, the LoRA series of techniques provides a concise and efficient solution for the fine-tuning of pre-trained language models in medium- to high-resource and even super-large-scale scenarios and show good application prospects in terms of multi-task reloadability and seamless integration of inference. With more in-depth research on intrinsic dimensionality and low-rank differences, Reparameterization-based PEFT will certainly continue to evolve in the future and bring more breakthroughs for the practical deployment of large-scale models and multi-task adaptation.

4 Experiment

Xu et al. (2023) [11] systematically evaluated the performance of multiple parameter efficient fine-tuning methods on the RoBERTa-base model. A series of experiments based on the

widely used natural language understanding evaluation benchmark GLUE (General Language Understanding Evaluation) were conducted to systematically evaluate the performance and resource utilization efficiency of different PEFT methods. GLUE consists of several subtasks, including syntactic judgment (CoLA), sentiment classification (SST-2), sentence-to-sentence equivalence judgment (MRPC, QQP), semantic similarity regression (STS-B), and natural language inference (MNLI, QNLI, RTE), etc., to comprehensively examine the generalization ability of the model in multiple language understanding scenarios. Each task is evaluated using Matthews Correlation, Accuracy, F1 Score, and Pearson/Spearman correlation coefficient to ensure that the model performance is quantified from multiple dimensions(Table 1).

Table 1 Comparison plot of efficient fine-tuning experiments for RoBERTa-base on the GLUE baseline [11].

Model	PEFT Method	#TPs	CoLA	SST2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	Avg
RoBERTa-base	FT	124.6M	59.07	92.89	88.24/91.58	90.87/90.61	90.81/87.72	86.27	91.01	72.20	84.00/84.00
	Adapter	7.41M	63.32	94.31	90.44/93.18	91.25/90.94	90.81/86.55	87.33	92.06	73.56	85.39/85.16
	Prompt-tuning	0.61M	49.37	92.09	70.83/81.72	82.44/83.11	82.99/78.35	80.57	80.03	58.12	74.56/75.42
	Prefix-tuning	0.96M	59.31	93.81	87.25/91.03	88.48/88.32	87.75/84.09	85.21	90.77	54.51	80.89/80.88
	BitFit	0.69M	61.32	94.72	89.22/92.41	90.34/90.27	88.12/84.11	84.64	91.09	77.98	84.68/84.57
	LoRA	0.89M	62.09	94.04	87.50/90.68	90.66/90.83	88.83/85.21	86.54	92.02	72.92	84.33/84.29
	AdaLoRA	1.03M	59.82	93.92	87.99/91.33	90.83/90.73	88.58/84.98	86.26	91.43	70.04	83.61/83.56

In the setting of evaluation metrics, Xu et al. chose the most representative performance measures according to the characteristics of each subtask to fully reflect the model's ability

in different language comprehension tasks. Specifically, for the syntactic judgment task CoLA, the Matthews Correlation Coefficient (MCC) was used, which is applicable to binary classification problems with category imbalance and can more accurately reflect the correlation between model predictions and real labels; the sentiment classification task SST-2 and the three natural language reasoning tasks (MNLI, Sentiment classification task SST-2 and three natural language reasoning tasks (MNLI, QNLI and RTE) use the most direct classification accuracy (Accuracy) as the evaluation criterion.

For the two tasks of sentence pair similarity judgment (MRPC and QQP), both accuracy and F1 score are reported, where the F1 score combines precision and recall, which is more suitable for dealing with label imbalance. In the semantic text similarity regression task STS-B, Pearson and Spearman correlation coefficients were used to measure the linear and ranked consistency between the model outputs and the manual scores, thus providing a more comprehensive assessment of the model's ability to model the semantic continuous space. Finally, the main evaluation metrics of each task are normalized, and their average (Avg.) is calculated as a measure of the overall performance. This indicator system has good representativeness and differentiation, providing a scientific and rigorous basis for comparing the performance of different PEFT methods.

To ensure the representativeness and comparability of the experiments, RoBERTa-base (125M parameters) is chosen as the base pre-training model, covering the current mainstream encoder-only architecture. In the selection of fine-tuning methods, the experiments cover several representative classes of PEFT methods, including Adapter for structure insertion, Prompt-tuning and Prefix-tuning for input guidance, BitFit for parameter subset updating, and LoRA based on low-rank parameterization and its variant, AdaLoRA. in addition, the full-volume fine-tuning for complete model parameter updating is also set up. Full Fine-Tuning (FT) for complete model parameter updates as a comparison baseline.

From the experimental results, all PEFT methods drastically reduce the number of trainable parameters, but there are significant differences in performance. The Full Fine-Tuning method uses all 124.6M parameters and achieves an average score of 84.00. In contrast, the Adapter method only fine-tunes about 7.41M parameters (about 5.9%), but achieves an average score of 85.39, and the overall performance surpasses that of full fine-tuning, and the performance is especially outstanding in the MRPC task, achieving the highest score of 90.44/93.18, which demonstrates a very high efficiency of parameter utilization. Under the extremely low parameter budget, the BitFit method only updates 0.69M bias parameters (<1%), but it performs stably in most tasks, with an average score of 84.68/84.57, and achieves the highest accuracy of 77.98 in the RTE classification task, showing its strong adaptability under low resource settings. Similarly, the LoRA method achieves an average performance of 84.33/84.29 with 0.89M parameter (about 0.7%), which is almost the same as full fine-tuning, while its variant AdaLoRA, while introducing dynamic rank control, achieves an average score of 83.61/83.56 with 1.03M parameter, which is slightly lower than LoRA but still maintains high efficiency.

In contrast, the Prompt-tuning method has the lowest number of parameters (only 0.61M), but the overall performance is significantly lower, with an average score of 74.56/75.42, especially in low-resource tasks such as RTE (with an accuracy of only 58.12), which shows a limited ability to migrate tasks. The Prefix-tuning method slightly outperforms Prompt-tuning, with an average score of 80.89/80.88 using the 0.96M parameter, which is a strategy with a better trade-off between efficiency and performance.

In conclusion, the experiment verifies that multiple PEFT methods can maintain or even surpass the performance of full fine-tuning in mainstream downstream tasks while significantly compressing the trainable parameters. In particular, methods like Adapter, BitFit, and LoRA show excellent performance and stability with a parameter ratio of less than 6% and display extremely high efficiency in parameter utilization. This provides an

important technical path and empirical support for future deployment of large-scale language models in resource-constrained environments.

5 Prospect

The efficient fine-tuning technique (PEFT) has become one of the key tools for downstream tasks of large model adaptation, but it still has some problems and challenges in practice, which limit its performance and wide application. The following discussion and outlook are based on both existing defects and future research directions.

5.1 Current drawbacks

First, PEFT techniques have the disadvantage of leading to letting models with insufficient robustness and generalization capabilities. Most of the current efficient fine-tuning methods (e.g., LoRA, Prefix Tuning, and Adapter) perform well in specific tasks, but often face performance degradation when migrating across tasks. This mainly stems from the fact that fine-tuning models are overfitted on specific data distributions and cannot adequately adapt to diverse scenarios. In addition, since fine-tuning methods usually introduce additional parameter modules, these modules may bring instability in dynamically changing environments and affect the robustness of the models.

Second, the problem of computational efficiency and resource consumption still exists. Although the PEFT technique significantly reduces the need for full-model parameter updating, its computational overhead is still not negligible when dealing with super-large-scale models. For example, methods such as LoRA introduce additional matrix operations in the multi-head attention mechanism, which may increase the memory consumption and computational latency. For resource-constrained devices or task scenarios, this extra overhead needs to be further optimized.

At the same time, PEFT technology lacks a unified theoretical framework and performance evaluation standards. The current PEFT methods are working on their own, and the unified design principles and theoretical guidance are not yet perfect. This not only makes it difficult to directly compare the advantages and disadvantages of different methods but also limits the collaboration and innovation of researchers in method improvement. In addition, the performance evaluation criteria are fragmented and lack common metrics for different tasks and scenarios.

Finally, privacy and security issues are also challenges that require researchers to apply them in practice. In the context of increasingly stringent data privacy and security requirements, PEFT methods need to achieve efficient adaptation without leaking the original data. However, existing methods have less research on coping with privacy leakage and model protection, and the support for sensitive data protection is still insufficient.

5.2 Prospects

In response to the above existing drawbacks and challenges, this paper proposes the following outlook for future research directions. First, researchers should consider designing stronger generalization ability and robustness mechanisms. In the future, a more generalized design of adaptation modules can be explored so that they can maintain stable performance in multi-tasks and multi-scenarios. For example, the adaptability of the model to different data distributions can be improved by introducing dynamic parameter tuning mechanisms or cross-task shared feature representations.

Second, applications need to further consider optimizing computational efficiency and hardware suitability. To address the need for efficient fine-tuning of large-scale models, research should be conducted on lightweight solutions that can further reduce memory consumption and computational complexity. For example, quantization or pruning techniques can be used to optimize the adaptation module of the model or develop a more adaptable architecture design for hardware gas pedals to meet the needs of resource-constrained scenarios.

Then, the construction of a unified theoretical framework and evaluation system is also what needs to be improved nowadays. A systematic comparison and analysis of existing PEFT methods should be conducted to refine a unified theoretical framework to help researchers better understand and improve the performance of the methods. At the same time, a common evaluation standard should be established to make the performance comparison more convincing and to promote the synergistic development of the field.

Finally, the practical application of how privacy protection and data security capabilities should be improved is also a key direction. With the support of technologies such as privacy computing and federated learning, research on technical solutions that can achieve efficient fine-tuning while protecting data privacy. For example, developing PEFT methods based on cryptographic computation, or exploring security adaptation strategies based entirely on model weight updates to meet the needs of sensitive data scenarios.

6 Conclusion

This paper summarizes the latest research progress of PEFT for LLMs. From the perspectives of computational efficiency, adaptation effect and practical application scenarios, it systematically analyzes the current mainstream PEFT technologies, including LoRA, Prefix Tuning and Adapter approaches. Focusing on the background of PEFT technology and its challenges in terms of computation cost and storage demand, this paper summarizes three major technology schools: additive-based, specific-based and reparameterization-based, and makes a generalized comparison with experimental verification and application scenarios.

By summarizing and analyzing the existing studies, this paper finds that PEFT technology can significantly reduce the cost of LLMs fine-tuning while maintaining a better adaptation effect in multi-tasking scenarios, which demonstrates its potential for wide application in natural language processing, computer vision and multimodal tasks. However, PEFT techniques still have obvious shortcomings in terms of model generalization ability, robustness, computational efficiency and privacy protection. For example, the performance degradation of current methods during cross-task migration needs to be addressed, while optimizations for hardware acceleration and resource-constrained scenarios need to be further explored. In addition, the existing PEFT methods have not yet formed a unified specification in terms of theoretical framework and evaluation criteria, which limits the comparison and collaboration among different methods.

In the future, the research of PEFT technology should focus on improving the model generalization ability and robustness, optimizing the computational efficiency, constructing a unified theoretical framework, and combining with the privacy protection technology to further promote its extensive landing in practical applications. With the continuous expansion of the scale of LLMs and the enrichment of application scenarios, PEFT technology will play a more important role in reducing the model training cost, improving the adaptation efficiency and promoting the development of AI. This paper hopes to provide valuable reference and inspiration for researchers in related fields.

References

1. T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell et al. "Language models are few-shot learners", Advances in Neural Information Processing Systems, vol. 33, Vancouver, Canada, 2020, pp. 1877–1901.
2. Y. Zhuang, Y. Yu, K. Wang, H. Sun, and C. Zhang "ToolQA: A dataset for LLM question answering with external tools", arXiv preprint, arXiv:2306.13304, 2023.
3. W. Zhu, H. Liu, Q. Dong, J. Xu, L. Kong, J. Chen, L. Li, and S. Huang "Multilingual machine translation with large language models: Empirical results and analysis", arXiv preprint, arXiv:2304.04675, 2023.
4. G. Li, H. A. A. K. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem "Camel: Communicative agents for mind exploration of large language model society", Thirty-seventh Conference on Neural Information Processing Systems, New Orleans, USA, 2023.
5. Z. Han, et al. "Parameter-efficient fine-tuning for large models: A comprehensive survey", arXiv preprint, arXiv:2403.14608, 2024.
6. Y. Li, et al. "Tokens-to-token ViT: Training vision transformers from scratch on ImageNet", IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, Canada, 2021.
7. K. Zhou, et al. "Learning to prompt for vision-language models", International Journal of Computer Vision, vol. 130, no. 9, 2022, pp. 2337–2348.
8. N. Ding, et al. "Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models", arXiv preprint, arXiv:2203.06904, 2022.
9. A. Vaswani, et al. "Attention is all you need", Advances in Neural Information Processing Systems, vol. 30, Long Beach, USA, 2017.
10. K. He, et al. "Deep residual learning for image recognition", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, USA, 2016.
11. L. Xu, H. Xie, Z. J. Qin, et al. "Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment", arXiv preprint, arXiv:2312.12148, 2023.
12. N. Houlsby, et al. "Parameter-efficient transfer learning for NLP", International Conference on Machine Learning (ICML), Long Beach, USA, 2019.
13. Y.-L. Sung, J. Cho, and M. Bansal "Lst: Ladder side-tuning for parameter and memory efficient transfer learning", Advances in Neural Information Processing Systems, vol. 35, New Orleans, USA, 2022, pp. 12991–13005.
14. X. L. Li and P. Liang "Prefix-tuning: Optimizing continuous prompts for generation", arXiv preprint, arXiv:2101.00190, 2021.
15. B. Lester, R. Al-Rfou, and N. Constant "The power of scale for parameter-efficient prompt tuning", arXiv preprint, arXiv:2104.08691, 2021.
16. J. Lee, R. Tang, and J. Lin "What would Elsa do? Freezing layers during transformer fine-tuning", arXiv preprint, arXiv:1911.03090, 2019.
17. E. Zaken, S. Ravfogel, and Y. Goldberg "BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language models", arXiv preprint, arXiv:2106.10199, 2021.

18. M. Raghu, et al. “Transfusion: Understanding transfer learning for medical imaging”, Advances in Neural Information Processing Systems, vol. 32, Vancouver, Canada, 2019.
19. A. Aghajanyan, L. Zettlemoyer, and S. Gupta ‘Intrinsic dimensionality explains the effectiveness of language model fine-tuning’, arXiv preprint, arXiv:2012.13255, 2020.
20. E. J. Hu, et al. “LoRA: Low-rank adaptation of large language models”, International Conference on Learning Representations (ICLR), Virtual Conference, 2022, pp. 1–3.