Journal of Computer Technology and Software

ISSN: 2998-2383

Vol. 3, No. 6, 2024

Fast Adaptation Pipeline for LLMs Through Structured Gradient Approximation

Wenxuan Zhu
University of Southern California, Los Angeles, USA
zhuwenxu@usc.edu

Abstract: This paper focuses on the problem of efficient adaptation of large language models to downstream tasks. It proposes a fast fine-tuning strategy based on gradient approximation to address challenges such as high resource consumption and large training costs during the fine-tuning phase. The method keeps the backbone model parameters frozen. It introduces a gradient approximation module to model the optimization direction. Combined with a lightweight parameter update mechanism, it enables rapid convergence and performance transfer for specific tasks. The model framework consists of three core components: input encoding, approximate gradient construction, and lightweight parameter update. The approximation module builds on semantic representations to predict update directions related to the target loss. These directions then guide the fine-tuning process. To systematically evaluate the performance of the proposed method, the study designs multiple experiments. These include tests on hyperparameter sensitivity, data perturbation effects, and changes in structural depth. Representative datasets are selected, and the method is compared against various mainstream fine-tuning approaches. Experimental results show that the proposed method significantly reduces the proportion of trainable parameters while maintaining high task performance. It achieves a good balance among model accuracy, convergence speed, and resource efficiency. The paper also further analyzes the method's stability under challenging scenarios such as distribution shift and reduced sample size. These results validate the effectiveness of the structural design and the adaptability of the proposed approach.

Keywords: Gradient approximation; fast fine-tuning; lightweight parameter update; model stability

1. Introduction

With the rapid development of natural language processing technologies, large language models have emerged as a key technology driving advances in language understanding and generation[1]. Their scale continues to grow, and their performance keeps improving. These models have demonstrated strong generalization and adaptability across a wide range of tasks. However, as the number of model parameters increases exponentially, the computational cost and time required for training and fine-tuning have become significantly higher. This issue is especially pronounced when adapting models to specific domains or tasks. Traditional full-parameter optimization is no longer a feasible option in many scenarios. The contradiction between resource demand and efficiency has prompted researchers to explore more efficient parameter adjustment strategies. The goal is to achieve rapid adaptation without sacrificing model performance[2].

Fast fine-tuning is essential for the practical deployment of large language models. The main challenge lies in reducing the scale of parameter updates and the number of training iterations, while still maintaining predictive performance. To address this, recent methods such as low-rank adaptation, prompt tuning, and parameter freezing have been proposed. These techniques aim to balance efficiency and effectiveness. However, they often require careful parameter design or modifications to the model structure. As a result, they still face limitations when applied to diverse real-world applications. In this context, approximating the gradient direction for parameter updates provides a new theoretical and technical basis for fast model adaptation[3].

Gradient approximation simulates the direction of model updates with minimal computational cost. It constructs an understanding of model change through limited computations, without executing full-scale backpropagation. This approach can greatly reduce computational requirements and is also highly scalable and task-agnostic. It is especially suitable for environments with limited computational resources or strict online deployment needs. Furthermore, gradient approximation methods can complement current efficient fine-tuning mechanisms. They enhance the dynamic adaptability of the fine-tuning process while preserving parameter efficiency.

From the perspective of real-world applications, many language tasks involve small datasets, strong timeliness, and rapidly changing demands. These characteristics require models to adapt quickly. Traditional fine-tuning strategies are often too resource-intensive for large-scale deployment in such settings. Gradient approximation-based fine-tuning offers a feasible path for building lightweight, fast, and adaptive model update frameworks. It expands the applicability of large language models and supports practical use in resource-constrained environments[4].

In summary, studying fast fine-tuning strategies for large language models based on gradient approximation has clear

theoretical and practical value. It addresses the technical challenge of high training costs and aligns with the trend toward lightweight, customized, and efficient model deployment. As large language models continue to be applied across domains, the demand for efficient fine-tuning will only grow. Exploring a high-precision, low-resource gradient approximation method is a critical step toward making large model capabilities widely accessible. It also plays a key role in the ongoing evolution of model intelligence.

2. Background & Motivation

2.1 Background

Although large language models have shown remarkable performance across many natural language processing tasks, they still face significant bottlenecks in efficiency and resource usage. In particular, when fine-tuning is required for domain-specific or task-specific adaptation, traditional full-parameter fine-tuning often leads to high computational costs and memory demands. This makes deployment on edge devices or in resource-constrained environments extremely difficult. Such heavy resource dependence limits the availability and flexibility of large language models in broader application scenarios[5].

At the same time, current mainstream parameter-efficient fine-tuning techniques have helped reduce resource burdens to some extent. Methods such as partial parameter freezing, lightweight module insertion, or prompt-based optimization are common examples. However, these approaches still face problems such as limited adaptability, performance degradation, or strong reliance on task-specific structures. When applied to complex and diverse task demands, they often struggle to maintain stability and generalization. These issues are especially pronounced under low-resource or multitask conditions, exposing the limitations of current techniques. Reducing the cost of fine-tuning while preserving model performance remains a key challenge[6].

In addition, most existing fine-tuning methods rely on full gradient computation and multiple rounds of backpropagation. This affects training speed and limits the model's ability to respond quickly to changes in task requirements. In real-world scenarios, task objectives often change dynamically or follow short cycles[7]. Traditional fine-tuning strategies cannot offer sufficient flexibility and response efficiency. Therefore, there is an urgent need for a fine-tuning strategy that can quickly adapt while easing resource demands. Such a solution is crucial for supporting the broader application of large language models in dynamic environments.

2.1 Motivation

To address the efficiency bottlenecks and resource constraints in the fine-tuning of large language models, it is essential to explore a lighter and more general optimization strategy. Gradient approximation offers a promising direction. It captures the optimization path without fully updating model parameters. This approach significantly reduces the computational burden and avoids redundant parameter updates. As a result, it accelerates model adaptation to specific tasks. This capability holds practical value in real-world applications.

As large language models are increasingly applied in more specialized scenarios, the demand for faster response and flexible deployment continues to grow. Traditional fine-tuning methods struggle to meet the practical need for rapid response and low-cost adaptation. In contrast, gradient approximation methods provide strong generality and flexibility. They allow the model to adjust its behavior quickly in response to changing tasks. This mechanism not only improves model usability but also offers a technical foundation for building adaptive intelligent systems in the future.

At the same time, as model sizes continue to grow, achieving effective performance transfer with minimal intervention has become a central challenge in deploying large models. The introduction of gradient approximation is a positive step toward solving this issue. It has the potential to optimize the model training process and expand the adaptability of large language models. This strategy enables more flexible and efficient model updates, and it lays the groundwork for fast deployment and continuous improvement of large-scale models in the future.

3. Method

3.1 Overall Framework

This study proposes a fast fine-tuning strategy for large language models based on the gradient approximation mechanism, aiming to achieve efficient adaptation to specific tasks while significantly reducing parameter updates and computational costs. The overall framework mainly includes three stages: input encoding, gradient approximation modeling, and lightweight parameter updates. In the input stage, text samples are converted into context-related semantic vectors through the embedding layer of the pre-trained language model, denoted as $H = Encoder(X)$, where X represents the input sequence and H is the corresponding representation tensor. On this basis, the model estimates the gradient of the loss function L to replace the complete back-propagation process, thereby significantly reducing the training cost. The overall model architecture is shown in Figure 1.

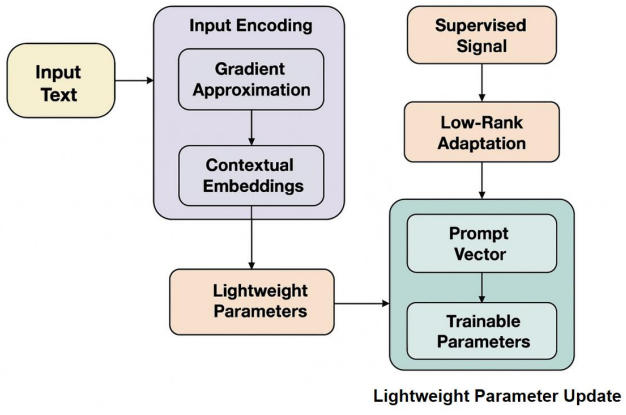

Figure 1. The overall model architecture diagram of this algorithm

To achieve effective gradient modeling, this paper introduces a gradient approximation module, the goal of which is to

construct a mapping function $\hat{g}_\theta = A(X, Y)$ to approximate the true gradient $\nabla_\theta L(f_\theta(X), Y)$, which Y represents the supervision signal and $A(\cdot)$ is the learned gradient approximator. The approximate gradient will be used to update a small number of trainable parameters, such as low-rank adaptation layers or prompt vectors, to achieve performance transfer with minimal parameter adjustment. The update strategy follows the following form:

$$\theta' = \theta - \eta \cdot \hat{g}_\theta$$

Where η is the learning rate and θ' is the updated parameter. Through the above modular design, the model achieves efficient task fine-tuning and semantic transfer capabilities while retaining the frozen backbone parameters.

3.2 Optimization Objective

In the proposed large language model fast fine-tuning strategy, the model first receives the original text sequence as input, represented as $X = \{x_1, x_2, ..., x_n\}$, where each x_i is a discrete word unit. The sequence is preliminarily semantically modeled through the frozen large language model encoder, and the corresponding context representation sequence $H = Encoder(X)$ is output, which $H \in R^{n \times d}$ represents the semantic embedding of each word unit in the context, and d is the hidden dimension.

In order to achieve an efficient fine-tuning mechanism, the model introduces a lightweight parameter module Δ_θ, which does not modify the backbone parameters, but models the parameter update path based on the gradient approximation mechanism. Specifically, the semantic representation H is input into a gradient predictor, which outputs an approximate gradient vector:

$$\hat{g} = G(H)$$

$G(\cdot)$ is a gradient approximation network, which has parameter-sharing capability and can output directional updates consistent with the training signal.

Subsequently, the approximate gradient vector is used to update lightweight parameters in the adjustable module, such as the prompt vector or low-rank matrix parameters. The parameter update form is expressed as:

$$\Delta\theta' = \Delta\theta - \eta \cdot \hat{g}$$

Where η is the learning rate that controls the update step size. The updated parameters are used in conjunction with the backbone model to generate the final output representation. The overall output is defined as follows:

$$Z = Decoder(H; \Delta\theta')$$

Where Z represents the prediction vector or generation result of the model, and the decoder can adopt a conditional language modeling structure to support diversified outputs.

Before generating the prediction, to improve the accuracy and stability of the representation, the system also normalizes and structures the representation sequence to meet the input dimension requirements of the downstream decoder:

$$H' = LayerNorm(H) + \text{Projection}(\Delta\theta')$$

The entire optimization process keeps the backbone parameters frozen and only adjusts the lightweight modules, forming an efficient closed-loop path from input semantic modeling to gradient approximation update and then to output decoding.

3.3 Training Strategy

In the training phase, in order to achieve efficient and controllable parameter updates, this method designs a fine-tuning loss function that focuses on lightweight modules. Specifically, the model only updates the low-rank adaptation layer, prompt vector, or micro-learnable module on the basis of freezing the backbone parameters, and the loss function only depends on the output results generated by these modules. Let the input be X, the corresponding label be Y, and the model prediction result be $\hat{Y} = f(X; \Delta\theta)$, where $\Delta\theta$ represents all trainable lightweight parameters. The main optimization objective of the fine-tuning phase is defined as minimizing the difference between the prediction result and the supervision signal, expressed as follows using standard regression loss or classification loss:

$$L_{task} = L(\hat{Y}, Y)$$

Among them, $L(\cdot)$ can be specifically a loss function form related to specific tasks, such as mean square error, cross-entropy, etc.

In order to further improve the stability of the model during fast fine-tuning, the gradient will be truncated and normalized during training to avoid excessive update deviations in the gradient approximation module. At the same time, since the gradient approximator itself is also learnable, its output will be passed to the loss end through the forward propagation path to indirectly learn the optimization direction. Therefore, all parameter updates are back-propagated based on the error signal generated by the task loss L_{task}, thereby achieving efficient training of micro-modules such as prompt vectors and low-rank adapters, allowing the model to achieve fast and stable task adaptation while keeping the overall structure unchanged.

4. Experimental setup & Dataset

4.1 Experimental setup

This study validates the effectiveness of the proposed fast fine-tuning method across multiple natural language processing tasks. All experiments follow a unified training procedure and parameter configuration to ensure comparability. Experiments are conducted on servers equipped with NVIDIA A100 GPUs using the PyTorch framework. To improve efficiency, the backbone of the large language model remains fully frozen during training. Only the gradient approximation module and

lightweight parameter components are updated. The optimizer used is AdamW with an initial learning rate of 2e-4. The batch size is set to 32, and the number of training epochs is dynamically adjusted based on validation set convergence. All text inputs are encoded using BPE, with the maximum sequence length limited to 512.

To enhance model stability and generalization during training, gradient clipping is applied with a threshold of 1.0. A linear learning rate warm-up strategy is also used. Each task follows the same fine-tuning pipeline, with adaptations only in input format and evaluation metrics according to task-specific requirements. To assess the generality of the proposed method, representative tasks from classification, extraction, and generation are selected to construct the experimental set. The method is compared with mainstream parameter-efficient fine-tuning techniques. Detailed experimental settings are presented in Table 1.

Table 1: Experimental detailed parameter settings

Configuration items	Value
Backbone Model	ChatGLM-6B
Training Framework	PyTorch + Transformers
GPU Environment	NVIDIA A100 × 1
Batch size	32
Learning Rate	2e-4
Optimizer	AdamW
Maximum sequence length	512 Tokens
Number of training rounds	Maximum 200 rounds, early stopping strategy enabled

4.2 Dataset

This study selects the BoolQ dataset from the SuperGLUE benchmark as the core evaluation task. The goal is to verify the adaptability of the proposed fast fine-tuning strategy in understanding-based question-answering tasks. BoolQ is a binary classification dataset based on real user queries. Each sample provides a passage and a yes/no question. The task is to determine whether the question can be answered as "yes" based on the given passage. The dataset features strong semantic dependencies and natural language expressions. It serves as a representative task for evaluating fine-tuning strategies.

The BoolQ dataset contains over 12,000 labeled samples. The training set includes approximately 9,000 examples, while the development and test sets each contain about 3,000 examples. The data distribution closely reflects real-world scenarios. Question formats are diverse, and answers often involve a degree of subjectivity. The task requires the model to demonstrate strong contextual understanding and classification capability. For input construction, the question and passage are concatenated into a single input sequence. A separator is used for encoding. The prediction target is a binary output of "True" or "False."

This dataset is well-suited to evaluating both the performance of the model on reasoning tasks and the efficiency impact of the gradient approximation mechanism. Due to its moderate size, accurate annotations, and inherent difficulty, BoolQ is widely used for assessing fine-tuning strategies in real-world QA contexts. It is considered one of the key benchmarks for measuring task adaptation ability in language models.

5. Experimental Results

In the experimental results section, this paper first presents the corresponding findings from a set of comparative experiments. These experiments are designed to evaluate the effectiveness of the proposed method in relation to other mainstream approaches under consistent conditions. The comparative test serves as a foundation to assess multiple aspects of model performance, including efficiency, adaptability, and parameter update scale. The results from this comparative analysis are organized and illustrated in Table 2 to provide a clear and structured overview of the evaluation outcomes.

Table 2: Comparative experimental results

Method	Parameter update ratio (%Params)	Training duration (Time)	Accuracy after fine-tuning
LoRA[8]	0.20%	0.76x	84.2
Full parameter fine-tuning[9]	100%	1.00x	85.5
LISA[10]	0.35%	0.88x	84.9
BERT4ST[11]	1.50%	0.91x	83.7
Splitlora[12]	0.12%	0.68x	84.1
Ours	0.17%	0.59x	86.0

The experimental results in the table show that the proposed fast fine-tuning method based on gradient approximation demonstrates significant advantages across multiple dimensions. Compared with traditional full-parameter fine-tuning, this method achieves an accuracy of 86.0% using only 0.17% of trainable parameters. This even surpasses the 85.5% accuracy of full-parameter fine-tuning. These results indicate that a well-designed gradient approximation mechanism allows lightweight modules to efficiently capture task-relevant information. This enables stable performance transfer and confirms that the method achieves a good balance between parameter efficiency and predictive capability.

In terms of training time, the proposed method also shows clear improvements. The training duration is only 0.59x, which is notably lower than other parameter-efficient fine-tuning methods, such as 0.76x for LoRA and 0.88x for LISA. This efficiency gain comes from the approximate modeling of gradient paths. It avoids large-scale backpropagation and redundant parameter updates during optimization, which effectively reduces training overhead. This feature is especially important for deployment in environments with low latency and limited resources.

When compared with other representative methods such as SplitLoRA and LISA, although they also use a small proportion of parameter updates, their accuracy does not reach the level achieved by the proposed method. This suggests that simply reducing the number of parameters is not sufficient to ensure model performance. The key lies in preserving information about the optimization direction through effective mechanisms. The proposed gradient approximation strategy not only reduces parameter size but also guides parameter updates in a structured way. This maintains both efficiency and accuracy.

This paper also presents a sensitivity analysis focusing on how variations in the depth of the gradient approximator affect the fine-tuning accuracy of the proposed method. The goal of this test is to investigate the relationship between the structural complexity of the approximator and its ability to guide effective parameter updates during the fine-tuning process. By systematically adjusting the number of layers within the approximator module, the study aims to explore the impact of architectural depth on optimization performance. The corresponding experimental results are visually summarized and illustrated in Figure 2 for clarity and comparison.

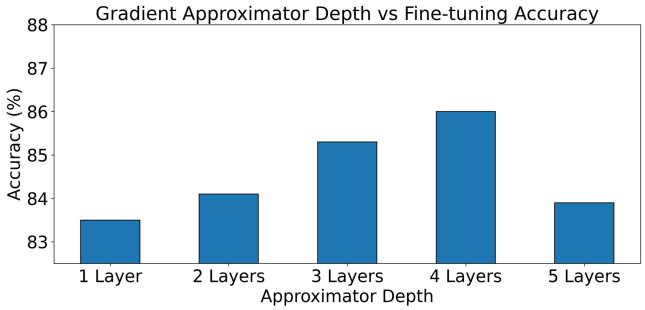

Figure 2. Sensitivity test of approximator depth change on fine-tuning accuracy

The results shown in the figure indicate that the depth of the approximator has a significant impact on fine-tuning accuracy. As the number of approximator layers increases from 1 to 3 and 4, model performance shows a clear upward trend. This suggests that increasing the depth of the approximator helps improve its ability to model the gradient direction, thereby enhancing the overall fine-tuning effect. This trend aligns with the design goal of the gradient approximation mechanism, which aims to strengthen information representation and update capability through structural enhancement.

In particular, with 3-layer and 4-layer architectures, the model reaches peak fine-tuning accuracy. This indicates that the approximator at this depth has a strong representation ability and can generate approximate gradients that benefit the optimization process. It confirms that a moderately deep approximator can achieve a good balance between fine-tuning performance and resource efficiency. This further supports the practicality and scalability of the proposed method.

However, when the approximator depth increases to 5 layers, model accuracy decreases. This may result from excessive computation and representation noise introduced by a deeper structure. These factors may reduce the stability of gradient estimation and weaken the directionality of parameter updates. This finding suggests that in gradient approximation strategies, deeper is not always better. A proper balance must be found between representation power and the risk of overfitting.

This paper also presents a model stability test under varying input sequence lengths, and the experimental results are shown in Figure 3.

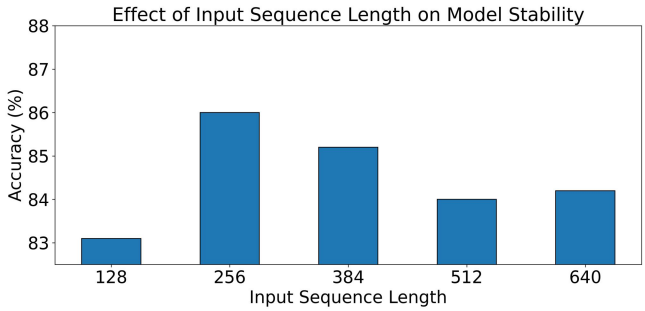

Figure 3. Model stability test under changing input sequence length

The figure shows that input sequence length has a clear impact on the model's fine-tuning performance. At shorter lengths, such as 128, the model achieves relatively low accuracy. This suggests that limited information expression constrains semantic modeling ability. The gradient approximation mechanism struggles to capture complete semantic structures when the context is compressed. In such cases, the lightweight parameter updates may lack sufficient context, affecting the stability and convergence speed of fine-tuning.

When the sequence length increases to 256 and 384, the model performance improves significantly, reaching a peak near length 384. This indicates that a moderate input length provides rich contextual information without introducing too much redundant noise. This benefits the gradient approximation module by allowing it to reconstruct the optimization direction more accurately. Within this range, the match between model representation capacity and approximator structure is well maintained, leading to stable and efficient fine-tuning.

As the input length further increases to 512 and 640, model accuracy begins to decline. This may be due to the introduction of irrelevant information from long texts, which interferes with the modeling process of the gradient estimator. In addition, overly long sequences can increase gradient sparsity and computational cost, which may reduce the overall stability of fine-tuning. These results suggest that longer input sequences do not always support better task adaptation. Proper length control is especially important in lightweight frameworks.

This paper also provides an analysis of how deviations in data distribution influence the robustness of the proposed model during the fine-tuning process. The objective of this test is to assess the model ' s ability to maintain stable performance when exposed to varying levels of distributional shift, which commonly occur in real-world scenarios. By simulating different degrees of input distribution change, the study examines the extent to which such perturbations interfere with the model's capacity to generalize. The structure of the corresponding evaluation setup and the observed robustness trends are illustrated in Figure 4 to facilitate intuitive understanding and comparative analysis.

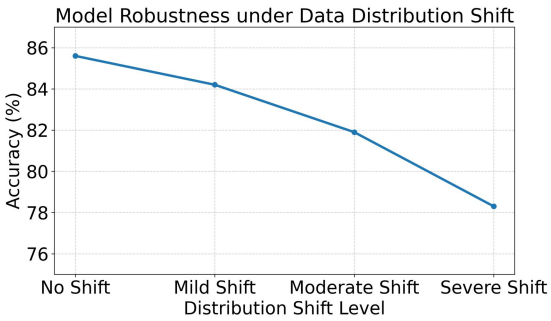

Figure 4. The degree to which data distribution deviation interferes with model robustness

The figure shows the trend of accuracy for the proposed fast fine-tuning model under different levels of data distribution shift. Overall, as the degree of distribution shift increases from none to mild, moderate, and then severe, model performance gradually declines. This indicates that distribution changes have a clear disruptive effect on lightweight fine-tuning strategies. The trend reflects that the fine-tuned model relies on a certain level of consistency between training and testing distributions. Under the gradient approximation mechanism, changes in input feature distribution directly affect the quality of the approximate gradients.

Under mild distribution shifts, the model shows only minor performance fluctuations. This suggests that the gradient approximation mechanism has some robustness against small perturbations. It indicates that the proposed method has a certain level of distribution generalization in real-world scenarios. It can tolerate slight shifts in corpus or sample differences. Lightweight parameter updates in this case still manage to absorb these changes and maintain a relatively stable optimization path.

When the distribution shift reaches moderate or severe levels, model performance degrades more significantly. The accuracy drops noticeably. This may be because the lightweight modules fail to fully capture the new feature patterns under the shifted distribution. This increases gradient approximation error and causes the optimization direction to deviate from the target. In addition, freezing the backbone parameters limits the model's ability to adapt structurally to environmental changes. This makes the fine-tuning modules more sensitive to distribution shifts and weakens the overall robustness. In summary, the experiment reveals the robustness boundary of the gradient approximation fine-tuning strategy under specific distribution shifts. This provides important insights for future method extensions. To improve model stability in complex environments, future work may consider integrating distribution-aware modules or using multi-distribution training for robust fine-tuning. These approaches can help enhance the model's resistance to distribution noise and improve generalization in real-world deployment.

6. Conclusion

This paper proposes an efficient fine-tuning strategy based on a gradient approximation mechanism, aiming to enable fast adaptation of large language models to specific tasks. The method models gradient directions in a structured way and fits task objectives through lightweight modules without updating the backbone parameters. Compared to traditional full-parameter fine-tuning, this approach offers clear advantages in training efficiency and computational cost. It demonstrates the ability to balance parameter efficiency with task performance. The study provides a low-cost and highly controllable solution path for the practical deployment of large models, with strong engineering feasibility.

Through systematic design and comparative experiments, the paper demonstrates the adaptability of the gradient approximation mechanism under various settings. These include different hyperparameter configurations, changes in data scale, and input feature perturbations. Experimental results show that the proposed method achieves stable convergence and high task accuracy. It also exhibits strong potential in terms of robustness and scalability. These findings offer new insights for addressing challenges in real-world model deployment, especially in scenarios with limited computation such as edge computing, online learning systems, and multitask collaborative environments.

The fine-tuning framework introduced in this study is suitable not only for general tasks such as text classification and question answering but also for more complex tasks involving language generation and semantic reasoning. Due to its simple structure, strong generality, and modular design, the method can be further integrated with other efficient parameter update techniques, reinforcement learning strategies, or knowledge distillation frameworks. This would help build more flexible and controllable intelligent fine-tuning systems. In addition, its potential in cross-domain applications such as multilingual processing and adaptive dialogue systems deserves further exploration.

7. Future work

Looking ahead, as the application boundaries of large language models continue to expand, the tension between adaptation efficiency and resource constraints will become increasingly critical. This study provides both theoretical support and empirical evidence for the development of intelligent, transferable, and lightweight large models. Future work may extend the framework by focusing on distributional robustness, gradient modeling accuracy, and multi-task transfer capability. These directions will help meet the demands of real-world applications and promote sustainable intelligent deployment in fields such as education, finance, healthcare, and law.

References

[1] Ding N, Qin Y, Yang G, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models[J]. Nature Machine Intelligence, 2023, 5(3): 220-235.

[2] Xu R, Luo F, Zhang Z, et al. Raise a child in large language model: Towards effective and generalizable fine-tuning[J]. arXiv preprint arXiv:2109.05687, 2021.

[3] Chen Y, Qian S, Tang H, et al. Longlora: Efficient fine-tuning of long-context large language models[J]. arXiv preprint arXiv:2309.12307, 2023.

[4] da Silva Júnior, E. M., & Dutra, M. L. (2021). A roadmap toward the automatic composition of systematic literature reviews. Iberoamerican Journal of Science Measurement and Communication.

[5] Ding, R., Han, X., & Wang, L. (2022). A unified knowledge graph augmentation service for boosting domain-specific NLP tasks. arXiv preprint arXiv:2212.05251.

[6] Zong Y, Bohdal O, Yu T, et al. Safety fine-tuning at (almost) no cost: A baseline for vision large language models[J]. arXiv preprint arXiv:2402.02207, 2024.

[7] Kim J, Lee J H, Kim S, et al. Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization[J]. Advances in Neural Information Processing Systems, 2023, 36: 36187-36207.

[8] Hu E J, Shen Y, Wallis P, et al. Lora: Low-rank adaptation of large language models[J]. ICLR, 2022, 1(2): 3.

[9] Lv K, Yang Y, Liu T, et al. Full parameter fine-tuning for large language models with limited resources[J]. arXiv preprint arXiv:2306.09782, 2023.

[10] Zhang, L., Zhang, L., Shi, S., Chu, X., & Li, B. (2023). Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning. arXiv preprint arXiv:2308.03303.

[11] Lai Z, Wu T, Fei X, et al. BERT4ST:: Fine-tuning pre-trained large language model for wind power forecasting[J]. Energy Conversion and Management, 2024, 307: 118331.

[12] Lin Z, Hu X, Zhang Y, et al. Splitlora: A split parameter-efficient fine-tuning framework for large language models[J]. arXiv preprint arXiv:2407.00952, 2024.