Optimizing Transformers and Large Language Models: Effectiveness Through Training and Fine-Tuning

Sameer Sundarrao Kakade[1], Pranav Shahaji Kamble[2], Dr. Prakash Kene[3]
[1,2,3]*MCA Department, PES Modern College of Engineering Pune, India*

Abstract- Transformer-based large language models have significantly advanced the field of natural language processing, delivering cutting-edge results across numerous tasks. However, as these models scale to billions of parameters, fully fine-tuning them becomes increasingly resource-intensive, both in terms of computation and storage. This paper investigates strategies to enhance the training and fine-tuning of transformers, with an emphasis on parameter-efficient approaches. Commonly known as "delta-tuning," these techniques allow models to be adapted to new tasks by updating only a minimal part of the parameters, yet still achieve performance close to that of full fine-tuning. This research shows recent developments in this area, examining their effectiveness, scalability, and relevance across a variety of NLP applications. By lowering the hardware and memory requirements, parameter-efficient tuning emerges as a practical and scalable method for refining LLMs, enabling broader accessibility and easier deployment across different sectors.

I. INTRODUCTION

Transformers, introduced by vaswani et al. In 2017 (attention is all you need), have become the foundation of modern nlp systems due to their ability to handle sequential data effectively through self-attention mechanisms. Large language models (llms) such as bert (bert: pre-training), gpt (improving language understanding), and t5 (exploring transfer learning) have demonstrated remarkable capabilities in understanding and generating human-like text, achieving state-of-the-art results in tasks like text classification, translation, and question answering. Nevertheless, the expanding size of these models, frequently comprising billions of parameters, presents substantial difficulties in the process of fine-tuning.

Refinement is an essential stage in tailoring pre-trained models to specific tasks, enabling them to utilize their broad knowledge base while focusing on particular applications. The conventional method of fine-tuning entails modifying all the parameters of the model, which is computationally demanding and necessitates significant storage space for each fine-tuned version. As llms become bigger, this method becomes impractical for many practitioners due to the high computational and memory requirements.

In order to tackle these difficulties, scientists have created efficient fine-tuning methods, which focus on adjusting only a small subset of the model's parameters while keeping the majority unchanged. These techniques, collectively known as "delta-tuning," strive to achieve performance similar to full fine-tuning while using fewer resources. This paper examines recent developments in parameter-efficient fine-tuning methods for language models, specifically analyzing their effectiveness, efficiency, and applicability in different natural language processing tasks.

II. LITERATURE REVIEW

2.1 Traditional Fine-Tuning

The process of fine-tuning llms entails training the entire model on a dataset tailored to a specific task, adjusting all parameters to minimise the task-specific loss. Although this method can result in exceptional performance, it is computationally intensive and demands significant storage space for each refined model version. For instance, refining a model like gpt-3 with 175 billion parameters demands substantial computational power, rendering it impractical for numerous applications (language models few-shot).

2.2 Parameter-Efficient Fine-Tuning

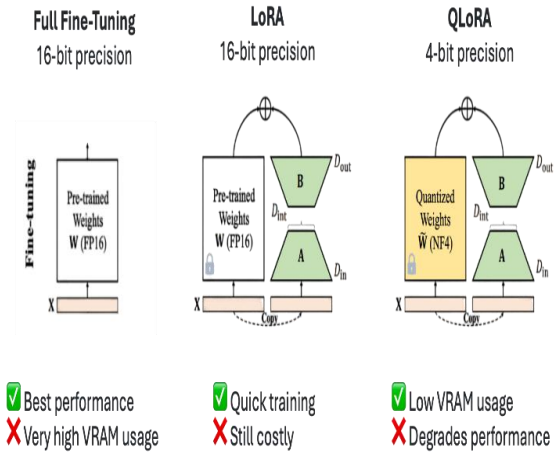
To address the limitations of fine-tuning, various ways have been developed that require fewer parameters. These techniques only focus on a limited set of parameters, known as "delta" parameters, while keeping the majority of the model unchanged. Key techniques include:

- prompt-tuning (pt): involves learning from a small set of prompt tokens that are input sequence prepended. The model is adjusted by focusing only on the prompt tokens while keeping the rest of the model unchanged (power of scale).
- prefix-tuning (pf): similar to prompt-tuning, but instead of learning prompt tokens, it learns a sequence of virtual tokens (prefix) that are prepended to the input at each layer (prefix-tuning)
- low rank adaptation (lora): decomposes the weight updates into low-rank matrices, reducing the number of trainable parameters (lora)
- adapter: inserts small, trainable modules (adapters) into each layer of the transformer, while keeping the original parameters frozen (parameter-efficient transfer)
- bitfit: fine-tunes only the bias terms of the model, which are a small fraction of the total parameters (bitfit)

These techniques have demonstrated promising outcomes in preserving performance while significantly decreasing computational and storage expenses.

2.3 Practical Implementation

Practical guides, such as those from hugging face (fine-tuning transformers) and datacamp (fine-tuning llms), emphasize the use of pre-trained models and tools like the hugging face trainer api for efficient fine-tuning. These resources emphasize effective techniques like data preprocessing, hyperparameter tuning, and regular evaluation to prevent problems like overfitting or underfitting.



Fine Tuning Techniques

III. METHODOLOGY

Each parameter-efficient fine-tuning method utilizes a distinct approach to optimize a specific subset of parameters.

- prompt-tuning: the model learns task-specific prompts that are concatenated with the input to guide the model's output. This technique is helpful for tasks where the input format can be altered.
- prefix-tuning: by learning layer-specific prefixes, method allows for more flexible adaptation across different layers of the transformer
- lora: factorizes weight updates into low-rank matrices, reducing the number of trainable parameters for instance, lora can significantly decrease the number of tunable parameters from 175 billion (in gpt-3) to just 37.7 million (lora).
- adapter: inserts small neural networks (adapters) between transformer layers These adapters acquire task-specific characteristics without modifying the initial model's weights.
- bitfit: updates only the bias terms, leveraging their significant influence on the model's behavior with minimal computational overhead

These techniques can be used separately or combined, depending on the task and model architecture. For example, combining adapters with other techniques can improve performance, but the best combination depends on the specific task.

3.1 Practical Techniques

Additional techniques for fine-tuning include:

- supervised fine-tuning: trains on labeled datasets for tasks like text classification or named entity recognition (fine-tuning llms)
- few-shot learning: uses few examples in prompts to improve context without extensive fine-tuning
- domain-specific fine-tuning: adjusts models to specific industries, such as medical or legal domains
- regularization: prevents catastrophic forgetting using techniques like elastic weight consolidation (ewc)

IV. RESULTS

4.1 Performance

Parameter-efficient fine-tuning techniques have demonstrated the ability to achieve similar performance levels to full fine-tuning across various

natural language processing (nlp) tasks. For example, ding et al. (parameter-efficient fine-tuning) evaluated over 100 nlp tasks and found that the average performance ranking was ft > lr > ap > pf > pt, with no single method consistently outperforming others. Nevertheless, larger models tend to derive greater advantages from these techniques. For example, prompt-tuning demonstrated enhanced performance when applied to larger models such as t5 large.

Performane Ranking

Method	Performance Ranking	Tunable Parameters (e.g., GPT-3)
Full Fine-Tuning	1st	175,255M
LoRA	2nd	37.7M
Adapter	3rd	Varies
Prefix-Tuning	4th	Varies
Prompt-Tuning	5th	Varies

4.2 Convergence

Delta-tuning techniques tend to progress at a slower pace compared to full fine-tuning, with the order being ft > ap ≈ lr > pf. Nevertheless, bigger models speed up convergence, indicating that the advantages of delta-tuning are more significant with larger parameter-efficient fine-tuning.

4.3 Efficiency

One of the main benefits of delta-tuning is its efficiency. It can save up to 75% of GPU memory when working with small batch sizes, such as 1 or 8, and up to 50%–66% when working with larger batch sizes, like 32 or 64. For instance, when using an Nvidia A100 GPU with 39.58 GB of memory, delta-tuning techniques compared to full fine-tuning (parameter-efficient fine-tuning). Furthermore, the time required for backpropagation is decreased because there are fewer adjustable parameters to consider.

Batch Size	Memory Savings
1, 8	Up to 3/4
32, 64	1/2–1/3

Memory Savings

4.4 Combinability

Mixing various delta-tuning techniques can occasionally improve performance, but the ideal combination varies depending on the specific task and model. For instance, combining adapters with other techniques frequently enhances performance, whereas prompt-tuning may occasionally lead to a decline in performance (parameter-efficient fine-tuning).

4.5 Power of Scale

Larger plms are more advantageous from delta-tuning. For example, with t5 xxl (11 billion parameters), techniques such as adapter, lora, and prefix-tuning can achieve performance that is very similar to full fine-tuning (parameter-efficient fine-tuning).

4.6 Transferability

Parameters that are tuned for one task can be applied to similar tasks, and text generation tasks have shown good transferability to sentiment analysis. This suggests that the learned adaptations can be applied to similar tasks (parameter-efficient fine-tuning).

4.7 Practical Example

A detailed example from datacamp (fine-tuning gpt-2 for tweet sentiment analysis) showcases the process of fine-tuning gpt-2 using the hugging face trainer API. The process entails loading a dataset, breaking it down into smaller units, setting up the model, and training it with specific settings, resulting in high accuracy with limited resources.

V. DISCUSSION

Parameter-efficient fine-tuning techniques represent a significant advancement in optimizing transformers and llms.Paragraph: Parameter-efficient fine-tuning techniques have emerged as a groundbreaking development in the field of transformer and llms optimization. These methods enable models to be easily adjusted for optimal performance on consumer devices, without requiring excessive computational resources. Each approach has its advantages: prompt-tuning and prefix-tuning are effective for tasks where input manipulation is advantageous, lora and adapters strike a balance between performance and efficiency, and bitfit offers a lightweight alternative.

Nevertheless, these techniques are not without drawbacks. For instance, prompt-tuning necessitates meticulous planning of prompts, and its effectiveness can be influenced by the selection of prompts. Furthermore, while delta-tuning can help reduce computational costs, it may not always provide the same level of performance as full fine-tuning, particularly in tasks that require extensive adaptation

from the pre-trained model. Practical challenges encompass preventing overfitting, underfitting, and catastrophic forgetting, which can be addressed through meticulous hyperparameter tuning and regularization (fine-tuning llms).

Future research could concentrate on refining combination strategies for delta-tuning methods, delving deeper into their theoretical underpinnings, and examining their effectiveness on multimodal models or in ongoing learning situations. Furthermore, examining the differences between fine-tuning and alternatives like retrieval-augmented generation (rag) could shed light on their respective advantages for various applications (rag vs fine-tuning).

VI. CONCLUSION

Parameter-efficient fine-tuning techniques provide a promising solution for enhancing the performance of transformers and large language models. These methods help practitioners reduce the computational and storage costs involved in fine-tuning llms, allowing them to adapt llms to specific tasks more efficiently. As llms continue to expand in size and significance, these techniques will be essential for their practical implementation and utilization. This review emphasizes the efficiency of delta-tuning methods, their ability to handle large-scale models, and their potential to make advanced nlp models accessible to a wider audience.

REFERENCE

[1] Attention is All You Need – arXiv, accessed on April 23, 2025.
[2] BERT: Pre-training of Deep Bidirectional Transformers – arXiv, accessed on April 23, 2025.
[3] Improving Language Understanding by Generative Pre-Training – OpenAI, accessed on April 23, 2025.
[4] LoRA: Low Rank Adaptation of Large Language Models – arXiv, accessed on April 23, 2025.
[5] Prakash Kene, Significance of financial planning and forecasting for Indian multinational companies, The Online Journal of Distance Education and e-Learning, Volume -11,issue-1,2023.
[6] Prakash Kene, Single Page Web Application Technologies, International Journal for Research in Applied Science & Engineering Technology, Volume -8,issue-5,2021.