TLoRA: Tri-Matrix Low-Rank Adaptation of Large Language Models

Tanvir Islam
Okta
Bellevue, WA
tanvir.islam@okta.com

Abstract

The Abstract paragraph should be indented ½ inch (3 picas) on both left- and right-hand margins. Use 10 point type, with a vertical spacing (leading) of 11 points. The word **Abstract** must be centered, bold and in point size 12. Two line spaces precede the Abstract. The Abstract must be limited to one paragraph.

1 Introduction

Fine-tuning is a critical process in the adaptation of large language models, aiming to tailor the model to perform specific tasks or solve specific problems (Chung et al. 2024; Naveed et al. 2023; Ouyang et al. 2022; Zhang et al. 2023). The technique involves adjusting the pre-trained model's weights by exposing it to task-specific data, thereby refining its understanding and responses based on the given context. This method leverages the foundational knowledge encoded within the model while introducing new patterns and nuances pertinent to the targeted problem statement.

Fine-tuning has garnered significant interest among practitioners and researchers working with large language models (LLMs) (Naveed et al. 2023; Wei et al. 2023). One common approach is full fine-tuning, which involves continued training of the model to specialize it for a specific task, such as sentiment analysis (Prottasha et al. 2022) or question answering (Luo et al. 2023), by using task-specific data. However, this method can be computationally expensive and time-consuming for LLMs. Alternatively, Parameter-Efficient Fine-Tuning (PEFT) presents a cost-effective solution. In PEFT, instead of updating all the model parameters, only a subset or additive parameters is adjusted. This selective fine-tuning reduces the computational overhead while still achieving high performance, making it an attractive option for resource and time-constrained environments.

Parameter-Efficient Fine-Tuning (PEFT) methods offer innovative approaches to adapt large pre-trained models with minimal computational overhead, crucial for resource-constrained environments (Ding et al. 2023; Han et al. 2024). PEFT methods can be broadly classified into three categories: additive methods, selective methods, and re-parameterized methods. Additive methods, such as Adapter Layers and Prompt Tuning, introduce additional trainable parameters into the model without modifying the original model's structure. These methods aim to efficiently learn task-specific information by appending new components that capture task nuances while preserving the base model. Selective methods, on the other hand, focus on optimizing a targeted subset of the model's existing parameters. By fine-tuning only selected layers or weights, these methods reduce computational cost while maintaining most of the model's original parameters. Finally, re-parameterized methods involve restructuring or transforming certain layers or components of the model in a way that changes how parameters are represented and optimized. This approach enables more efficient adaptation by embedding task-relevant information directly into modified parameterizations, often leading to better performance for domain-specific

applications while keeping the number of trainable parameters low. Together, these three approaches offer a spectrum of strategies for fine-tuning large models with minimal resource demands, each balancing model performance and computational efficiency differently.

Low-Rank Adaptation, or LoRA is one such method that has significantly advanced the field of Parameter-Efficient Fine-Tuning (Hu et al. 2021). In LoRA, certain layers of the model, typically within dense or attention layers, are re-parameterized by introducing low-rank matrices that effectively reduce the dimensionality of the weight updates needed for fine-tuning. Instead of updating the full set of parameters, LoRA learns a low-rank decomposition of parameter updates, which minimizes the number of trainable parameters while still allowing the model to capture task-specific nuances.

Inspired by the success of LoRA, we ask: Can we do even better? This paper introduces TLoRA: Tri-Matrix Low-Rank Adaptation of Large Language Models, a novel technique that leverages a tri-matrix structure along with a clever adaptive scaling mechanism. This approach not only maintains the efficiency of low-rank adaptation but also enhances the model's ability to capture complex interactions and nuances in data. By exploring this innovative method, we aim to push the boundaries of efficient model fine-tuning, delivering superior performance with reduced resource requirements.

2 TLoRA

2.1 Objective functions

We know that a pre-trained autoregressive language model $P_\theta(y \mid x)$ parameterized by θ can be adapted for various downstream tasks by fine-tuning it on task-specific data. In these tasks, we generally use a training dataset of context-target pairs $\mathcal{Z} = \{(x_i, y_i)\}_{i=1,\ldots,N}$, where consists of a sequence of tokens and y_i could either be a sequence of tokens corresponding to the answer generated by the model or a classification target, depending on the task. For example, in a natural language inference (NLI) task from the GLUE benchmark, x_i represents a pair of sentences (e.g., a premise and a hypothesis), and y_i is the corresponding classification label. The LLM model is then updated to optimize its performance for the specific task.

If we opt for full fine-tuning, the pre-trained model parameters θ_0 are updated directly to $\theta_0 + \Delta\theta$ by maximizing the conditional language modeling objective. In the context of classification tasks, such as those in the GLUE benchmark, we seek to optimize the model to predict a discrete class label y based on an input sequence x. Therefore, the objective becomes:

$$\max_\theta \sum_{(x,y)\in Z} \log P_\theta(y|x),$$

where $P_\theta(y|x)$ is the probability distribution over class labels given the input x, and θ represents the full set of model parameters (including the pre-trained ones θ_0 and the task-specific updates $\Delta\theta$.

However, this full fine-tuning approach is computationally expensive, especially for large models with billions of parameters like GPT-3 and Llama.

To address this issue, low-rank adaptation technique is a more efficient alternative where only a small set of task-specific parameters ϕ is learned. Instead of updating θ_0 directly, TLoRA uses a low-rank update $\Delta\theta(\phi)$, which is much smaller than the original parameter set θ_0. The optimization problem then becomes:

$$\max_\phi \sum_{(x,y)\in Z} \log P_{\theta_0+\Delta\theta(\phi)}(y|x),$$

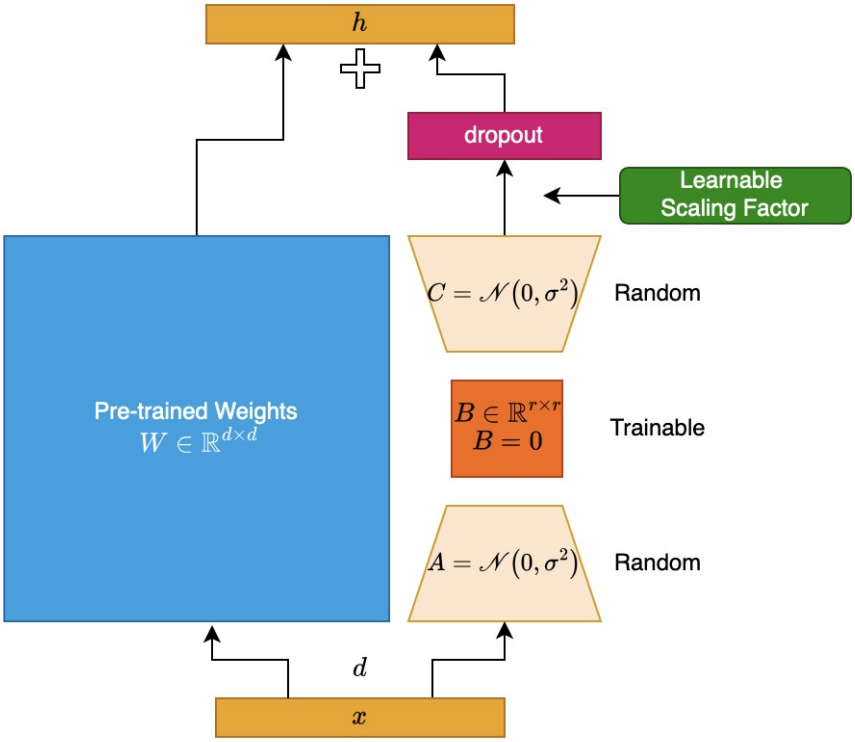
where $\Delta\theta(\phi)$ is the task-specific parameter update encoded by ϕ, and $P_{\theta_0+\Delta\theta(\phi)}(y|x)$ is the probability distribution over class labels for the adapted model.

Similar to LoRA, in TLoRA, the update $\Delta\theta(\phi)$ is represented in a low-rank form to make the adaptation both memory- and computation-efficient. By restricting the update to a low-rank representation, we ensure that the number of learnable parameters is much smaller than the size of the original model .

In this setting, the model's classification objective remains the same as in full fine-tuning, but now, instead of optimizing all of , we only need to optimize the much smaller parameter set , which encodes the low-rank adaptation.

2.2 Tri-matrix decomposition and adaptive scaling

TLoRA is an extension of the Low-Rank Adaptation (LoRA) technique that uses a tri-matrix decomposition for more flexible and efficient adaptation of pre-trained language models to downstream tasks. TLoRA allows for a more granular adaptation of the model's weights while maintaining a minimal increase in parameters by introducing three low-rank matrices. Additionally, TLoRA incorporates a trainable scaling factor to control the magnitude of the low-rank update. TLoRA's fine-tuning mechanism is illustrated in Figure X.



TLoRA Decomposition and Weight Update

Given a pre-trained autoregressive language model with weight matrix $W_0 \in \mathbb{R}^{d \times k}$, TLoRA computes the weight update ΔW through a tri-matrix decomposition. Specifically, we decompose the update into three low-rank matrices:

$$A \in \mathbb{R}^{k \times r}, \quad B \in \mathbb{R}^{r \times r}, \quad C \in \mathbb{R}^{r \times d}$$

where r is the rank of the decomposition, typically much smaller than d ( ) to ensure that

the number of parameters added during adaptation remains small. The original weight matrix W_0 of the layer remains frozen.

In TLoRA, only the matrix B is trainable, while matrices A and C are randomly initialized and fixed (non-trainable) throughout adaptation. This design leverages the efficiency of low-rank updates without requiring significant parameter growth. The technical rationale for keeping A and C fixed and random is to create a structured yet efficient transformation that enhances model flexibility with minimal additional parameters. By making B trainable, TLoRA allows the model to learn a task-specific transformation within a constrained low-rank space defined by A and C.

The low-rank update is computed as the product of these three matrices:

The adapted weight matrix is then the sum of the original weight matrix W_0 and the low-rank update ΔW:

$$W_{\text{adapted}} = W_0 + \alpha \Delta W = W_0 + \alpha ABC$$

where $\alpha \in \mathbb{R}$ is a trainable scaling factor that controls the contribution of the low-rank adaptation.

In the forward pass of the TLoRA layer, the model performs the following steps. The input $x \in \mathbb{R}^d$ is passed through the original pre-trained linear transformation represented by W_0:

$$h_0 = W_0 x$$

Here, x is the input (e.g., a token embedding), and $h_0 \in \mathbb{R}^d$ is the output of the standard linear transformation.

The low-rank adaptation is computed by multiplying the input x with the tri-matrix decomposition:

$$\Delta h = \alpha \cdot (xABC)$$

Dropout is then applied to the low-rank update to regularize the model:

$$\Delta h_{\text{dropout}} = \text{Dropout}(\Delta h)$$

The final output of the TLoRA layer is the sum of the standard linear transformation and the low-rank update with dropout:

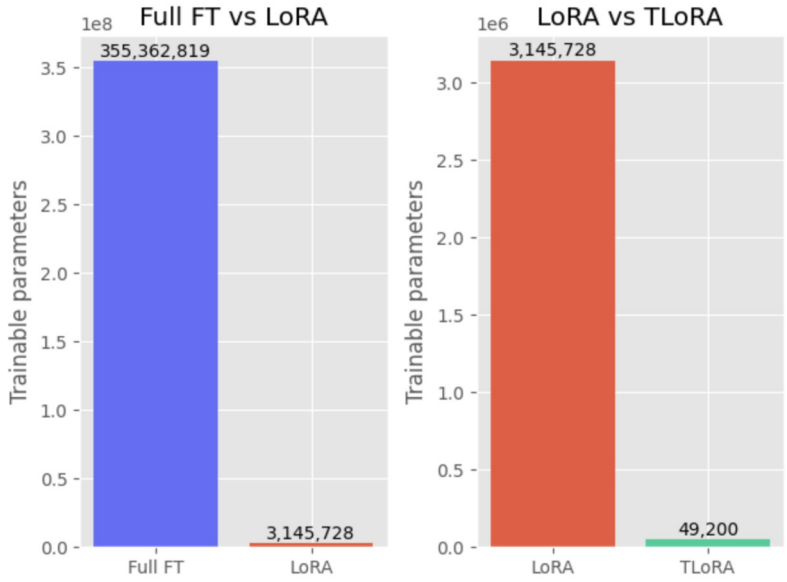
$$h = h_0 + \Delta h_{\text{dropout}} = W_0 x + \alpha \cdot (xABC)$$

Trainable Scaling Factor

The scaling factor α in TLoRA is a trainable parameter, which allows the model to dynamically adjust the contribution of the low-rank adaptation during training. This means that unlike in the original LoRA where α is determined through a fixed hyperparameter, in TLoRA, α is learned during training, allowing the model to adjust the strength of the low-rank update dynamically. This trainable scaling factor enables the model to fine-tune the balance between preserving the pre-trained knowledge and incorporating task-specific adaptations. During training, α is optimized through gradient descent along with other model parameters. By learning the optimal scaling factor for the target layer, TLoRA can flexibly control the strength of the low-rank update, allowing for more precise adaptation to specific tasks without introducing excessive complexity.

2.3. Parameter count

In TLoRA, we achieve significant parameter reduction compared to conventional fine-tuning and LoRA while maintaining similar adaptation capacity. For example, in a model like RoBERTa large with 355 million parameters, a standard fine-tuning approach requires training all parameters, resulting in a high computational burden and storage cost. In contrast, LoRA achieves a reduction by introducing only low-rank matrices of rank r, where the trainable parameter count is proportional to r. For instance, with $r = 8$, LoRA requires 786,432 trainable parameters, and this count increases with the increase in r, reaching 3,145,728 trainable parameters for $r = 32$.



TLoRA further optimizes parameter efficiency by leveraging a tri-matrix decomposition where only the middle matrix B is trainable, reducing the parameter count even more. This structure results in TLoRA needing only 3,120 trainable parameters at $r = 8$, representing a 252x reduction in trainable parameters compared to full fine-tuning. As r increases, TLoRA maintains substantial parameter savings over LoRA, with a 128x reduction at $r = 16$ and a 64x reduction at $r = 32$. A visual illustration is shown in Figure X.

Table X:

Rank	Full FT # Trainable Parameters	LoRA # Trainable Parameters	TLoRA # Trainable Parameters	Parameter Improvement by same rank	Parameter Improvement by (TLoRA r=32/LoRA r=8)
8	355362819	786432	3120	252x	
16		1572864	12336	128x	
32		3145728	49200	64x	16x

Notably, even at a higher effective rank (e.g., TLoRA with $r = 32$, TLoRA achieves a parameter count improvement of 16x over LoRA at $r = 8$, demonstrating TLoRA's ability to match LoRA's performance with significantly fewer trainable parameters, enhancing memory efficiency and enabling scalability for large language models on parameter-constrained hardware. The detailed comparison of parameters count of TLoRA to LoRA is shown in Table X.

2.4. Initialization

In TLoRA, non-Trainable Matrices A and C are initialized with a Kaiming Normal distribution. The trainable matrix $B \in \mathbb{R}^{r \times r}$ is initialized to zeros. This ensures that the low-rank update $\Delta W = ABC$ contributes no additional transformation at the start of training, allowing the model to preserve its pre-trained behavior. The scaling parameter α is initialized to 1.0, allowing a balanced initial contribution of the low-rank update. This initialization ensures that TLoRA begins training from a stable configuration, effectively leveraging the representational capacity of A and C while dynamically learning task-specific transformations in B. We also apply a dropout rate of 50% to the low-rank update ΔW to prevent overfitting and improve generalization.

3 Empirical experiments

We hypothesize that the weight updates required for fine-tuning large pre-trained language models have a low "intrinsic rank," meaning that task-specific adaptations can be captured effectively within a smaller, structured subspace. Prior work supports this idea, showing that language models can achieve efficient learning even when projected into lower-dimensional spaces. TLoRA capitalizes on this by introducing a tri-matrix decomposition, which enables richer, more expressive representations in low-rank adaptations. In TLoRA, the weight update is decomposed into three matrices—two fixed random matrices on the input and output dimensions, with only the middle matrix being trainable—enabling the model to capture subtle, task-specific transformations without inflating the parameter count.

Moreover, TLoRA incorporates a learnable, layer-specific scaling factor to dynamically adjust the contribution of the low-rank update at each layer. This adaptive scaling factor enhances the model's flexibility, allowing it to vary the influence of the low-rank transformation in each layer based on the specific demands of the task. This added flexibility, combined with the tri-matrix structure, helps TLoRA to efficiently capture complex input interactions and enables fine-grained control of the adaptation strength throughout the network. By dynamically adjusting the influence of the low-rank components, TLoRA provides an effective balance between parameter efficiency and expressive power, making it well-suited for parameter-constrained applications while preserving adaptation quality across a range of NLP tasks.

In this study, we evaluate the effectiveness of TLoRA on several downstream tasks, comparing it against LoRA in terms of performance and parameter efficiency. Can TLoRA perform competitively with LoRA, despite having a fewer number of parameters? We focus on four classification tasks from the GLUE benchmark: MRPC (Microsoft Research Paraphrase Corpus), RTE (Recognizing Textual Entailment), QNLI (Question-answering NLI), and SST-2 (Stanford Sentiment Treebank). These tasks span a range of natural language understanding challenges, from sentence similarity to sentiment analysis, offering a comprehensive evaluation of the models' capabilities.

We conduct our experiments using the pre-trained transformer model RoBERTa-large, a bidirectional encoder model designed for robust language understanding tasks. Specifically, we leverage the MNLI checkpoint, which has been fine-tuned on the Multi-Genre Natural Language Inference (MNLI) dataset. This choice aligns with prior work, allowing for a direct comparison of our proposed TLoRA method with existing adaptation techniques under consistent conditions. The MNLI-tuned RoBERTa-large provides a strong baseline, as it is optimized for handling complex sentence-pair classification tasks, making it well-suited for evaluating the effectiveness of low-rank adaptation methods like TLoRA.

To perform fine-tuning, we apply TLoRA to specific layers of the model. We target the linear layers associated with the attention mechanisms and intermediate feed-forward layers, as these are the most critical for task adaptation. For RoBERTa, the model is composed of stacked transformer encoder layers, and we specifically target the linear layers within the attention submodules (self.query, self.key) and the output projection layers, as these play a central role in the model's ability to capture semantic relationships between tokens. For OPT, a decoder-only model, we

similarly target the linear layers in the decoder's attention mechanism, specifically focusing on the query (self_attn.q_proj) and value projections (self_attn.v_proj) within the self-attention submodules. These projections are essential for learning the relationships between the query and value vectors in the attention mechanism, which is central to autoregressive language modeling.

For each task, we fine-tune the models using both TLoRA and the standard LoRA approach. In TLoRA, we augment the low-rank updates by introducing an additional very-low-rank matrix and an adaptive scaling factor, which allows for a more flexible low-rank approximation of the parameter updates. This extension aims to improve the model's performance without introducing a significant increase in the number of trainable parameters. In contrast, LoRA only employs a single low-rank decomposition for the adaptation.

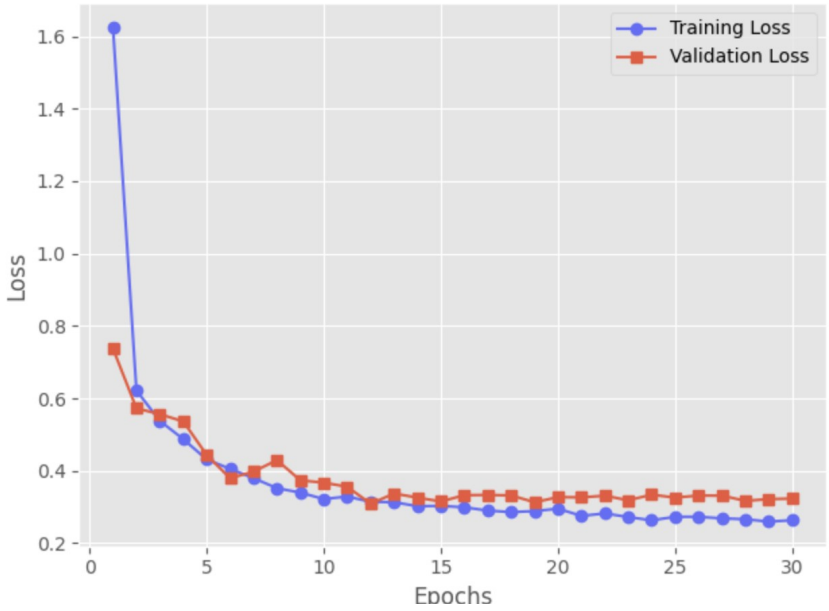
The objective of our experiments is to evaluate how well these adaptations—LoRA and TLoRA—perform on classification tasks, both in terms of classification accuracy and computational efficiency. We measure the impact of these techniques on the GLUE tasks by comparing the models' accuracy across all tasks, as well as the number of parameters that need to be fine-tuned.

	SST-2	**CoLA**	**QNLI**	**RTE**	**MRPC**
Batch Size	32	32	32	32	32
Leaning Rate	0	0	0	0	0
Epochs	20	20	20	20	20
Rank	32	32	32	32	32
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
LR Schedule	Linear	Linear	Linear	Linear	Linear

Our experimental setup, summarized in Table X, includes fine-tuning RoBERTa-large on five benchmark datasets: SST-2, CoLA, QNLI, RTE, and MRPC. For each dataset, we maintain consistent hyperparameters to ensure fair comparisons across tasks. We use a batch size of 32 and fine-tune for 20 epochs per task. The low-rank adaptation rank is fixed at 32 for all experiments to balance adaptation expressiveness with parameter efficiency. Optimization is performed using the AdamW optimizer, with a linear learning rate schedule applied to adjust the learning rate over training. This unified setup allows us to systematically evaluate the effectiveness of TLoRA in low-rank fine-tuning across diverse NLP tasks.

4 Results

First, we present the training and validation loss curves for TLoRA, shown in Figure X. The figure is constructed for the MRPC dataset over 30 epochs. TLoRA demonstrates stable training dynamics, with both training and validation losses decreasing consistently throughout the epochs before reaching a plateau. This stability highlights TLoRA’s ability to maintain effective learning with low-rank parameterization, avoiding issues such as overfitting or loss divergence. The close alignment between training and validation loss curves further illustrates TLoRA's generalization capability, suggesting that our tri-matrix decomposition and adaptive scaling techniques successfully capture task-relevant patterns without excessive parameter overhead.



In our experiments, TLoRA demonstrates competitive performance across multiple datasets while maintaining an exceptionally low parameter footprint. As shown in Table Y, TLoRA achieves an average accuracy of 81.48% across the SST-2, QNLI, RTE, and MRPC benchmarks with only 0.049M trainable parameters—significantly fewer than methods such as Adapter (AdptP) and LoRA, which use parameter counts ranging from 0.8M to 6M. Notably, TLoRA achieves a high 87.5% accuracy on the RTE task, outperforming larger configurations like AdptH with 0.8M parameters and approaching the performance of full fine-tuning methods. This parameter efficiency can be attributed to TLoRA's tri-matrix decomposition and adaptive scaling, which allow the model to capture complex task-specific information with minimal trainable weights. Although TLoRA's accuracy on certain datasets, such as QNLI, is slightly lower than other methods, its trade-off in parameter efficiency and competitive accuracy across tasks showcases its potential as a highly scalable, efficient adaptation technique for large language models.

Model	**Fine Tuning Method**	**Trainable Params**	**SST-2 (Accuracy)**	**QNLI (Accuracy)**	**RTE (Accuracy)**	**MRPC (Accuracy)**	**Avg**
RoBERTa	AdptP	3M	96.1	94.8	83.8	90.2	86.64
	AdptP	0.8M	96.6	94.8	80.1	89.7	85.8
	AdptH	6M	96.2	94.7	83.4	88.7	85.9
	AdptH	0.8M	96.3	94.7	72.9	87.7	83.58
	LoRA-FA	3.7M	96	94.4	86.1	90	86.9
	LoRA	0.8M	96.2	94.8	85.2	90.2	86.92
	VeRA	0.061M	96.1	94.4	85.9	90.9	87.06
	TLoRA	0.049M	95.3	92.1	87.5	89.3	81.48

5 TLoRA resembles LoRA

Electronic submissions are required, via this submission website:

6 Conclusion

Building on the foundational work of Low-Rank Adaptation (LoRA), we introduce TLoRA, a novel fine-tuning technique designed to enhance model adaptability and performance with computational efficiency.

TLoRA introduces a tri-matrix decomposition to adapt pre-trained language models, utilizing the matrices \(A \), \(B \), and \(C \) to compute a low-rank update to the model's weights. The contribution of this update is controlled by a **trainable scaling factor** \(\alpha \), which is learned during training. This approach strikes a balance between computational efficiency and adaptation flexibility, enabling effective model fine-tuning with a minimal increase in parameters. By using non-trainable matrices \(A \) and \(C \) and allowing \(B \) to be trainable, TLoRA enables efficient adaptation while maintaining the core functionality of the original pre-trained model.

By integrating these components, TLoRA offers a robust and efficient method for fine-tuning large language models, pushing the boundaries of model adaptability and performance. This novel approach showcases the potential for extending low-rank adaptation techniques to achieve greater efficiency and effectiveness in the evolving landscape of large language models.

References

Chung, Hyung Won, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, et al. 2024. “Scaling Instruction-Finetuned Language Models.” *Journal of Machine Learning Research* 25(70): 1–53.

Ding, Ning, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, et al. 2023. “Parameter-Efficient Fine-Tuning of Large-Scale Pre-Trained Language Models.” *Nature Machine Intelligence* 5(3): 220–35.

Han, Zeyu, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. “Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey.” *arXiv preprint arXiv:2403.14608*.

Luo, Haoran, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, et al. 2023. “Chatkbqa: A Generate-Then-Retrieve Framework for Knowledge Base Question Answering with Fine-Tuned Large Language Models.” *arXiv preprint arXiv:2310.08975*.

Naveed, Humza, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2023. “A Comprehensive Overview of Large Language Models.” *arXiv preprint arXiv:2307.06435*.

Prottasha, Nusrat Jahan, Abdullah As Sami, Md Kowsher, Saydul Akbar Murad, Anupam Kumar Bairagi, Mehedi Masud, and Mohammed Baz. 2022. “Transfer Learning for Sentiment Analysis Using BERT Based Supervised Fine-Tuning.” *Sensors* 22(11): 4157.

Wei, Fusheng, Robert Keeling, Nathaniel Huber-Fliflet, Jianping Zhang, Adam Dabrowski, Jingchao Yang, Qiang Mao, and Han Qin. 2023. “Empirical Study of LLM Fine-Tuning for Text Classification in Legal Document Review.” In *2023 IEEE International Conference on Big Data (BigData)*, IEEE, 2786–92.

Zhang, Shengyu, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, et al. 2023. “Instruction Tuning for Large Language Models: A Survey.” *arXiv preprint arXiv:2308.10792*.

lora_B mediates the interaction: The matrix lora_B can be thought of as the matrix that mediates the interaction between lora_A and lora_C in the low-rank update. If the norm of lora_B is large, the interaction between lora_A and lora_C is strong, meaning the low-rank adaptation could have a more significant effect on the final output. Scaling lora_B helps control this interaction and adjust the magnitude of the low-rank update.