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

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

Abstract

We propose **TLoRA**, a novel tri-matrix low-rank adaptation method that decomposes weight updates into three matrices: two fixed random matrices and one trainable matrix, combined with a learnable, layer-wise scaling factor. This tri-matrix design enables TLoRA to achieve highly efficient parameter adaptation while introducing minimal additional computational overhead. Through extensive experiments on the GLUE benchmark, we demonstrate that TLoRA achieves comparable performance to existing low-rank methods such as LoRA and Adapter-based techniques, while requiring significantly fewer trainable parameters. Analyzing the adaptation dynamics, we observe that TLoRA exhibits Gaussian-like weight distributions, stable parameter norms, and scaling factor variability across layers, further highlighting its expressive power and adaptability. Additionally, we show that TLoRA closely resembles LoRA in its eigenvalue distributions, parameter norms, and cosine similarity of updates, underscoring its ability to effectively approximate LoRA's adaptation behavior. Our results establish TLoRA as a highly efficient and effective fine-tuning method for LLMs, offering a significant step forward in resource-efficient model adaptation.

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. Foundational models, such as GPT (Brown 2020), BERT (Kenton and Toutanova 2019), Llama (Touvron et al. 2023a), and RoBERTa (Y. Liu 2019), have been pre-trained on vast corpora and capture a rich representation of language, including syntax, semantics, and general world knowledge. These models act as a robust starting point, offering a versatile understanding of language that can be fine-tuned for various downstream tasks, including natural language understanding and generation.

In recent years, there has been significant interest among practitioners and researchers in exploring various fine-tuning methods for 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. This method, while effective, can be computationally intensive for LLMs. On the contrary, Parameter-Efficient Fine-Tuning (PEFT) presents an alternative cost-effective solution. In fact, PEFT method is found to be better than in-context learning (H. Liu et al. 2022). In the 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; Xu et al. 2023). PEFT methods can be taxonomically divided into three primary paradigms: additive, selective, and re-parameterized. Additive methods, such as Adapter Layers and Prompt Tuning, introduce additional trainable parameters into the model without modifying the original model's structure (X. Wang, Aitchison, and Rudolph 2023). 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 (Han et al. 2024). 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 (Chen et al. 2024) 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. There are also variants of LoRA reported in the literature (Lialin et al. 2023; Kopiczko, Blankevoort, and Asano 2023; Li, Han, and Ji 2024).

Inspired by the promising results achieved by LoRA (Hu et al. 2021), we ask: Can we do even better? Can we further reduce the number of trainable parameters while maintaining the similar performance? 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. TLoRA, introduced in this paper, offers a more flexible and efficient approach for adapting large language models to diverse tasks. 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 $P_\theta(y|\,x)$ 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 $Z = \{(x_i, y_i)\}_{\mathrm{i=1,\dots,N}}$, where x_i 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 (Nangia and Bowman 2019), 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 (Floridi and Chiriatti 2020) and Llama (Touvron et al. 2023b). 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.

Like the 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 $|\phi|$ is much smaller than the size of the original model $|\theta_0|$. In this setting, the model's classification objective remains the same as in full fine-tuning, but now, instead of optimizing all of θ, we only need to optimize the much smaller parameter set ϕ, which encodes the low-rank adaptation.

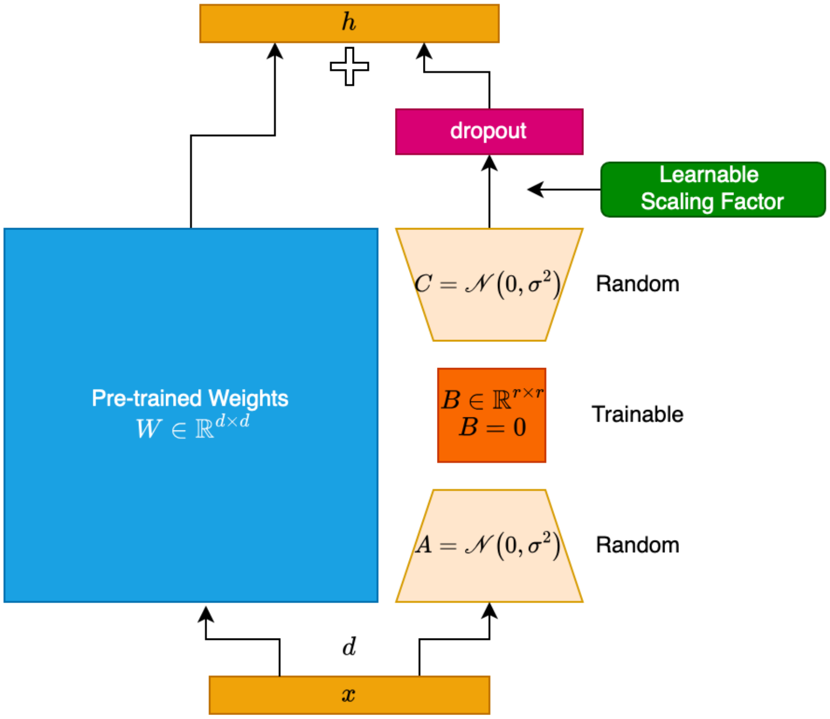
h
dropout
Learnable Scaling Factor
$C = \mathcal{N}(0, \sigma^2)$
Random
Pre-trained Weights
$W \in \mathbb{R}^{d\times d}$
$B \in \mathbb{R}^{r\times r}$
$B = 0$
Trainable
$A = \mathcal{N}(0, \sigma^2)$
Random
d
x

Figure 1: Schematic representation of TLoRA. The weight update is decomposed into a tri-matrix structure consisting of two fixed random matrices A and C, and a trainable matrix BBB. The input x is projected through the sequence of matrices A, B, C, followed by a learnable scaling factor to control the magnitude of the adaptation. This tri-matrix design enables efficient low-rank adaptation while minimizing trainable parameters.

2.2 Tri-matrix decomposition and adaptive scaling

TLoRA is an extension of the Low-Rank Adaptation (LoRA) technique (Hu et al. 2021) 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. Figure 1 provides a visual representation of TLoRA's fine-tuning process.

TLoRA Decomposition and Weight Update

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

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

where r is the rank of the decomposition, typically much smaller than d, ($r \ll d$) 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:

$$\Delta W = A\ B\ C$$

The adapted weight matrix $\mathrm{W}_{\mathrm{adapted}}$ is then the sum of the original weight matrix W_0 and the low-rank update ΔW:

$$\mathrm{W}_{\mathrm{adapted}} = \mathrm{W}_0 + \alpha\Delta\, W = \mathrm{W}_0 + \alpha\, ABC$$

where α 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 $x \in R^d$ 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 $h_0 \in R^d$ 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 (Y. Liu 2019) 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 $r = 32$.

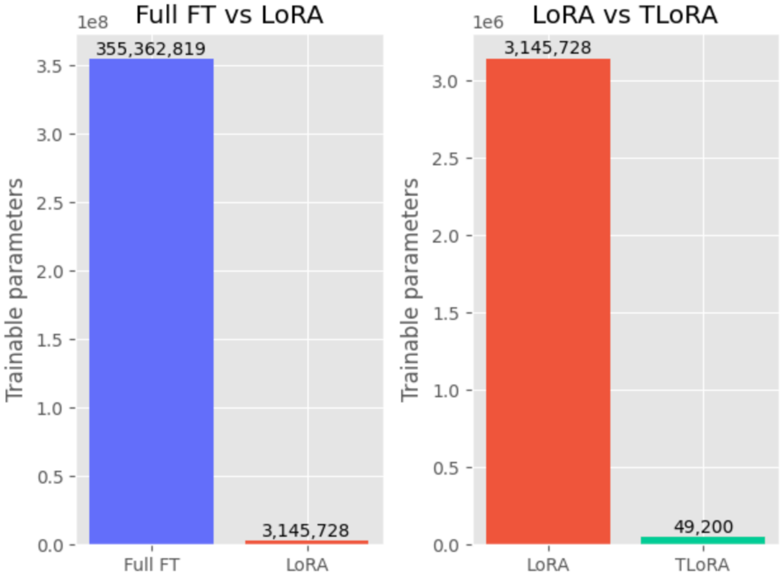

Figure 2: Comparison of trainable parameters across different fine-tuning methods. The bar chart

illustrates the parameter count for full fine-tuning (Full FT), LoRA (rank r=32), and TLoRA (rank r = 32). TLoRA significantly reduces the number of trainable parameters compared to LoRA, showcasing its parameter efficiency while maintaining competitive performance.

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 2 for r = 32.

Table 1: Theoretical trainable parameters between LoRA and TLoRA.

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 1.

2.4. Initialization

In TLoRA, non-Trainable Matrices A and C are initialized with a Kaiming Normal distribution (He et al. 2015). The trainable matrix $B \in R^{r \times r}$ is initialized to zeros. This ensures that the low-rank update $\Delta W = A\ B\ C$ 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.

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 (Hu et al. 2021; Lialin et al. 2023; Li, Han, and Ji 2024). 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 evaluate our approach on four classification tasks selected from the GLUE benchmark (A. Wang 2018): 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 (Matena and Raffel 2022; Hu et al. 2021). 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, 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.value) and the output projection layers, as these play a central role in the model's ability to capture semantic relationships between tokens. 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 TLoRA model. In particular, 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, we know that 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.

Table 2: TLoRA training parameters and setup for the GLUE tasks.

	SST-2	QNLI	RTE	MRPC
Batch Size	32	32	32	32
Leaning Rate	0	0	0	0
Epochs	30	30	30	30
Rank	32	32	32	32
Optimizer	AdamW	AdamW	AdamW	AdamW
LR Schedule	Linear	Linear	Linear	Linear

Our experimental setup, summarized in Table 2, includes fine-tuning RoBERTa-large on four benchmark datasets: SST-2, 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 30 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. We also apply a dropout rate of 50% to the low-rank update ΔW to prevent overfitting and improve generalization. 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 3. 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.

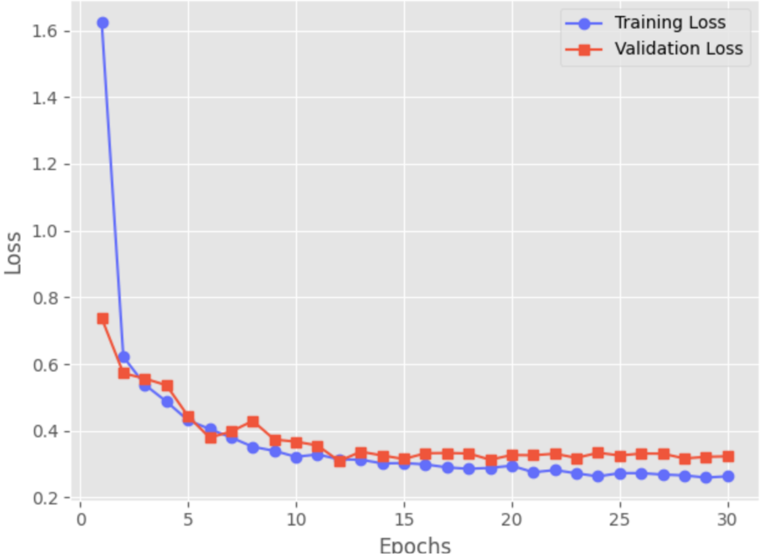

Figure 3: Training and validation loss curves for the MRPC dataset. The figure demonstrates the stability of TLoRA during training over 30 epochs.

In our experiments, TLoRA demonstrates competitive performance across multiple datasets while maintaining an exceptionally low parameter footprint. As shown in Table 3, 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. despite using over 16x fewer trainable parameters

Table 3: Results for different adaptation methods on the GLUE benchmark. Higher values indicate better performance for all metrics. The TLoRA results are based on our implementation, while the results for other methods are sourced from prior work (Hu et al. 2021; Kopiczko, Blankevoort, and Asano 2023).

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	91.2
	AdptP	0.8M	96.6	94.8	80.1	89.7	90.3
	AdptH	6M	96.2	94.7	83.4	88.7	90.8
	AdptH	0.8M	96.3	94.7	72.9	87.7	87.9
	LoRA-FA	3.7M	96	94.4	86.1	90	91.6
	LoRA	0.8M	96.2	94.8	85.2	90.2	91.6
	VeRA	0.061M	96.1	94.4	85.9	90.9	91.8
	TLoRA	0.049M	95.3	92.1	87.5	89.3	91.0

5 TLoRA adaptation dynamics

In this section, we provide an in-depth analysis of the behavior and adaptation dynamics of TLoRA; and compare its properties with those of LoRA to better understand its low-rank adaptation capabilities. For demonstration, we analyze the TLoRA-trained model on the MRPC dataset.

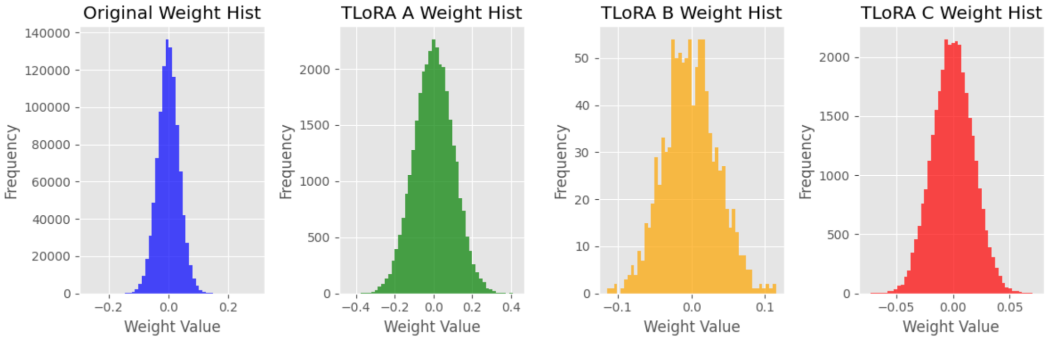

Figure 4: Weight distribution histograms for the original weight matrix and the TLoRA matrices A, B, and C. The matrices A and C are randomly initialized and remain fixed, following a Gaussian distribution. In contrast, the trainable matrix B, which is initialized to zero, evolves during training and adopts a Gaussian-like distribution, highlighting the effectiveness of TLoRA's tri-matrix decomposition.

5.1. TLoRA parameter behavior

We first examine the weight distributions of TLoRA’s tri-matrix components, as depicted in Figure 4. This is constructed for layer 0 and the query (q) matrix. we find clear evidence that the adaptation aligns well with natural statistical properties and maintains stability across updates. Matrices A and

C are randomly initialized using a Gaussian distribution to provide a stable, structured space for the low-rank adaptation. This initialization allows A and C to effectively capture diverse input-output interactions while remaining fixed throughout training. In contrast, matrix B, which is initialized to zeros, evolves significantly over training. Post-training, the weight distribution of B closely resembles a Gaussian pattern, indicating that it has learned a meaningful structure aligned with task-specific representations.

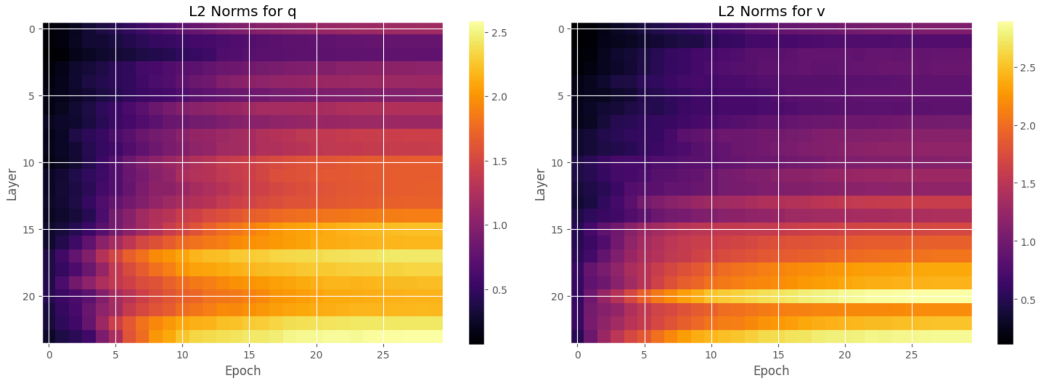

Figure 5: Evolution of L2 norms for the TLoRA B matrix over training epochs.

The evolution of TLoRA's L2 norms across training epochs provides further insights into the layer-wise adaptation dynamics, as shown in Figure 5. At the start of training (epoch 1), the L2 norms of TLoRA parameters for both the query and value matrices are close to zero, aligning with the initial zero-initialization of matrix B in TLoRA. This zero starting point ensures that TLoRA begins with minimal influence on the pre-trained model's output, allowing for a smooth and gradual adaptation to the target task as training progresses.

As training advances, we observe that the L2 norms increase, but the growth rates vary significantly across layers. This variation suggests that each layer's query and value projections adapt differently, likely reflecting the differential relevance of specific layers to the given task. The increase in L2 norm values with training indicates that TLoRA is progressively capturing task-specific information, with each layer's TLoRA parameters contributing to a learned adaptation in the low-rank subspace. This dynamic growth in the L2 norms supports our hypothesis that TLoRA efficiently leverages the minimal trainable matrix B to encode essential adaptations without excessive parameter overhead. The pattern observed in the L2 norms confirms that TLoRA is effectively engaging in layer-wise task-specific learning, adapting the low-rank representations as needed for each layer while maintaining efficient parameter utilization. This controlled and progressive adaptation process underscores TLoRA's capability to balance parameter sparsity with effective learning, a key objective in the design of our tri-matrix structure.

In examining the evolution of TLoRA's learned scaling factors over epochs, we observe substantial layer-wise variability, as depicted in Figure 6. Initially, each scaling factor starts at 1, providing a uniform impact across layers. However, as training progresses, we see distinct divergence in these factors: some layers show a significant increase in scaling values, while others decrease, reflecting a dynamic, adaptive adjustment based on each layer's contribution to task performance. Furthermore, within each layer, scaling factors differ between the query (q) and value (v) matrices, suggesting that TLoRA is refining its adaptation granularity to capture the unique functional roles of q and v within each layer.

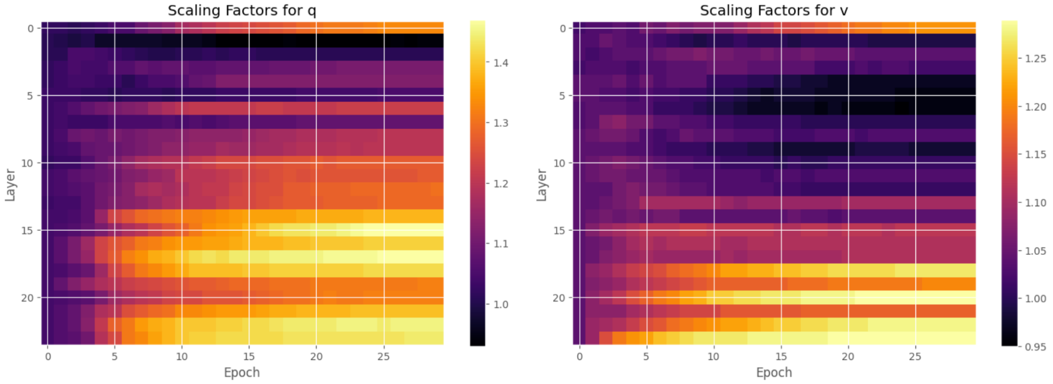

Figure 6: Evolution of scaling factors over training epochs for TLoRA. The scaling factors, which are learnable and layer-specific, start from an initial value of 1 and exhibit significant variability across layers.

This learned variability in scaling factors is a crucial component of TLoRA's adaptation mechanism. By allowing scaling factors to adjust independently across layers and between q and v, TLoRA dynamically modulates the influence of low-rank updates, enhancing the model's ability to capture task-specific nuances without disrupting pre-trained knowledge. Higher scaling factors in certain layers, for instance, indicate that those layers require more pronounced adaptations, which TLoRA accommodates by increasing the influence of its low-rank component in those areas. Conversely, reduced scaling factors in other layers suggest that minimal alteration is necessary, enabling TLoRA to selectively constrain changes where the pre-trained parameters are already well-aligned with the task.

This layer- and component-wise adaptability confirms TLoRA's capacity for flexible, fine-grained adjustment, allowing the model to engage in task-specific tuning without excessive parameter growth. The learned scaling factors thus act as a refined control mechanism, optimizing the integration of TLoRA's low-rank components and contributing to its effectiveness as an efficient yet expressive fine-tuning approach.

Figure 7 presents a heatmap of the layer normalization values for the query (q) matrices across the tri-matrix components A, B, C. As expected, since matrices A and C are randomly initialized and fixed throughout training, we observe minimal to no variability in their layer normalization values, yielding uniform heatmap patterns. This stability reflects that A and C retain their initialized structure and act as static projections, providing stable directions in the parameter space.

In contrast, matrix B, which is trainable, exhibits clear variability across layers in its layer normalization values. This variability indicates that B adapts dynamically during training, with each layer capturing distinct task-specific transformations. Interestingly, the layers with higher scaling factors, as previously discussed, also show correspondingly higher layer normalization values for B. This relationship suggests that the layers requiring larger adaptations to align with task demands exhibit more pronounced changes in B, which are reflected both in their scaling factors and in their layer normalization profiles.

These findings reinforce the role of B as the adaptable core of TLoRA's low-rank adaptation framework, responding layer by layer to task-specific requirements while leveraging the stable structures provided by A and C. The alignment between scaling factors and layer normalization values further highlights TLoRA's ability to target specific layers for adaptation intensity, enhancing both its flexibility and efficiency in fine-tuning transformer models.

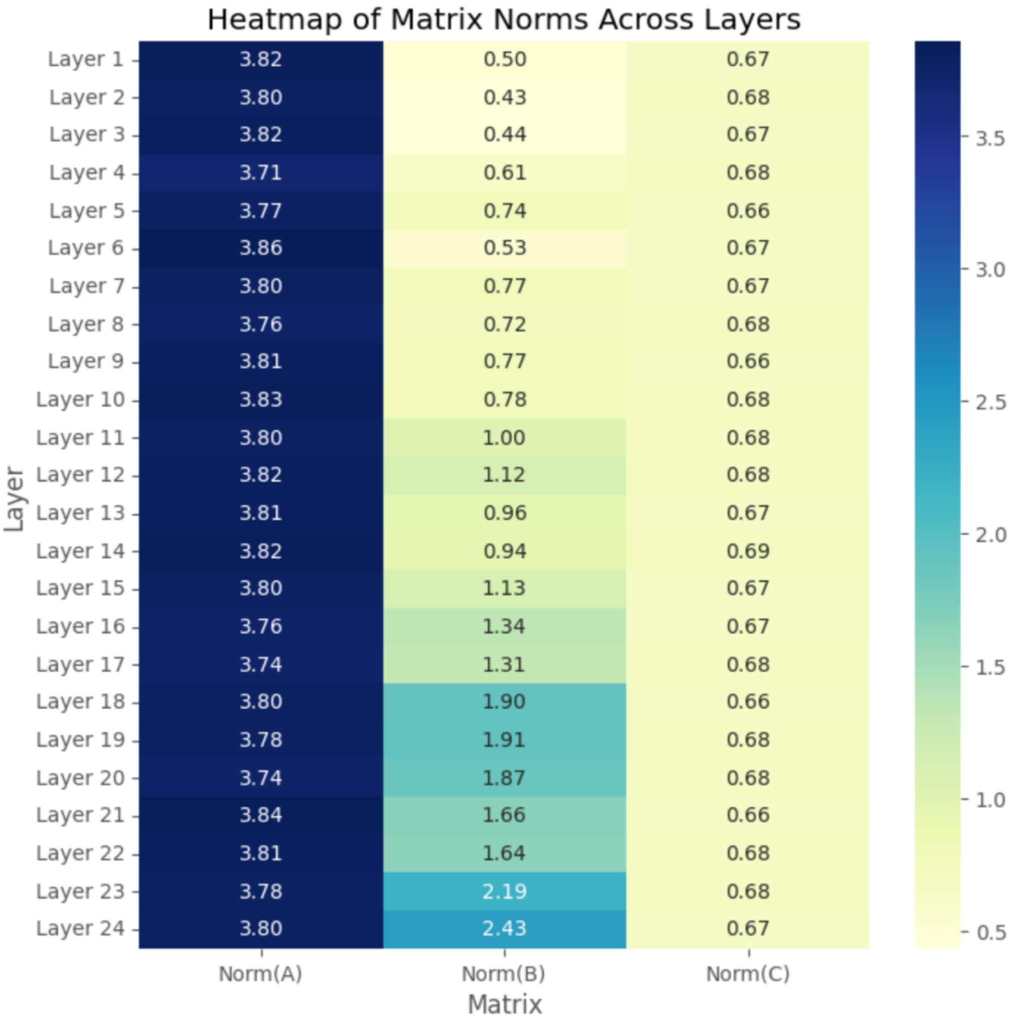

Figure 7: Heatmap of layer normalization values for the query (q) matrices across the tri-matrix components A, B, C. The fixed matrices A and C show no variability in their layer normalization values, as expected. In contrast, the trainable matrix BB exhibits significant variability across layers, which corresponds to the learned task-specific adaptations.

5.2. TLoRA resembles LoRA

To investigate the similarity between TLoRA and LoRA, we analyze whether TLoRA follows a comparable update trajectory to that of LoRA during training. Using the same experimental procedure as outlined in the LoRA paper, we first train a LoRA model. We subsequently evaluate the adaptation dynamics of both LoRA and TLoRA methods on the GLUE MRPC dataset, providing a detailed comparison of their behavior and performance during fine-tuning.

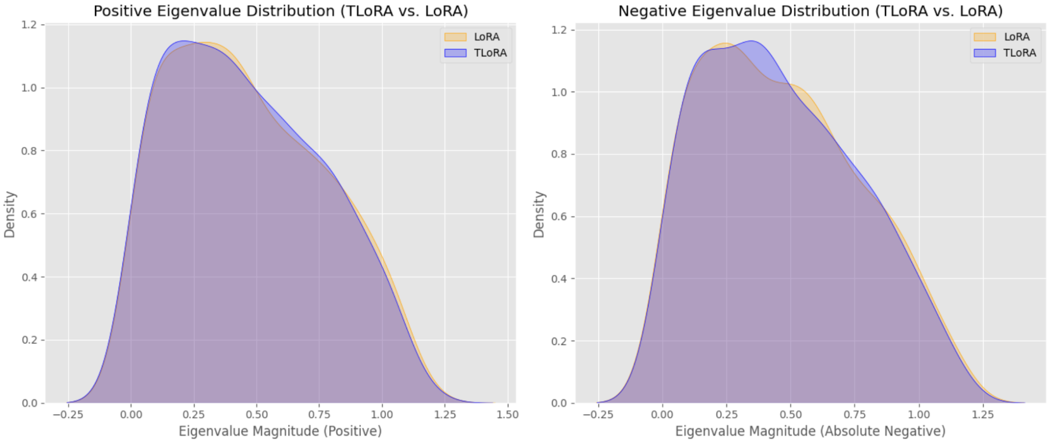

Figure 8: Eigenvalue distributions of the learned parameter updates for TLoRA and LoRA. The figure compares the eigenvalue distributions of the updates for both methods, showing that the distributions closely align in terms of both positive and negative eigenvalues.

In Figure 8, we present the eigenvalue distributions of the learned parameter updates for both LoRA and TLoRA, distinguishing positive and negative eigenvalues by their absolute values. Notably, the distributions align closely between the two methods, with TLoRA's eigenvalues closely mirroring the spread and magnitude of LoRA's. This similarity in eigenvalue behavior suggests that, despite TLoRA's use of a tri-matrix decomposition and fewer trainable parameters, it is able to capture the essential directions of adaptation in a manner analogous to LoRA. The resemblance in eigenvalue distributions underscores that TLoRA achieves an effective and efficient approximation of LoRA's adaptation pathway. This observation supports the hypothesis that TLoRA's tri-matrix design, even with fixed parameters in A and C, can reach a comparable low-rank solution, capturing the core task-specific transformations in a similarly structured manner.

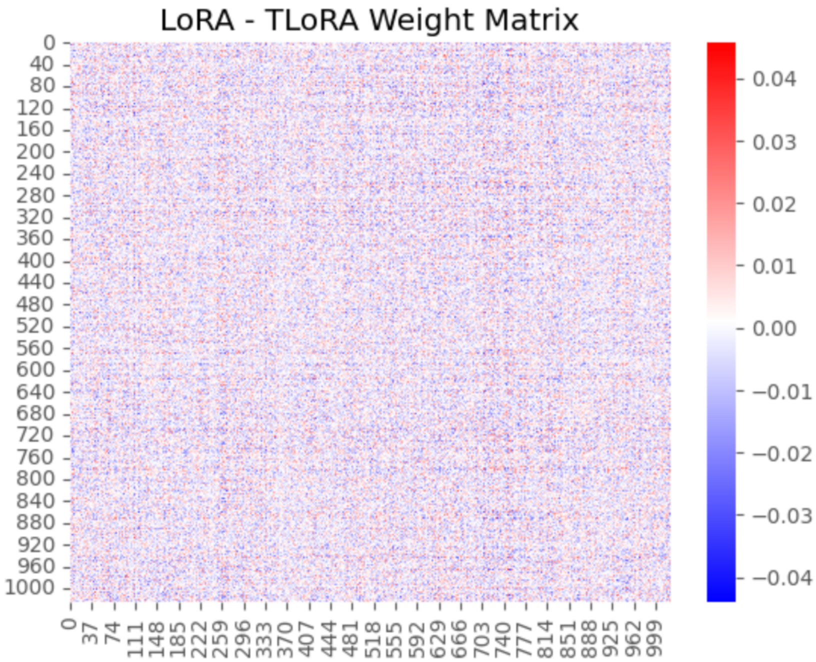

Figure 9: The figure shows the element-wise difference of the weight matrices between LoRA and TLoRA for a specific layer. The differences are close to zero, indicating that the adaptation behavior of TLoRA closely resembles that of LoRA.

This resemblance between TLoRA and LoRA is further highlighted by examining the element-wise differences between their adaptation matrices. In Figure 9, we visualize the difference between the LoRA and TLoRA matrices for the query (q) component in the first layer (layer 0). As shown, the values in the LoRA–TLoRA matrix are close to zero across most elements, indicating minimal divergence between the two methods in their parameter updates at this layer.

The near-zero differences suggest that, despite structural distinctions and TLoRA's added scaling and fixed matrices A and C, both methods arrive at similar low-rank parameter adjustments for this layer. This outcome further supports the idea that TLoRA effectively mirrors LoRA's adaptation without requiring identical parameter configurations, demonstrating that TLoRA can approximate LoRA's solution while leveraging its tri-matrix design. This similarity highlights TLoRA's ability to capture the essential adaptation directions in a computationally efficient manner, reinforcing its utility as a low-rank alternative for efficient model fine-tuning.

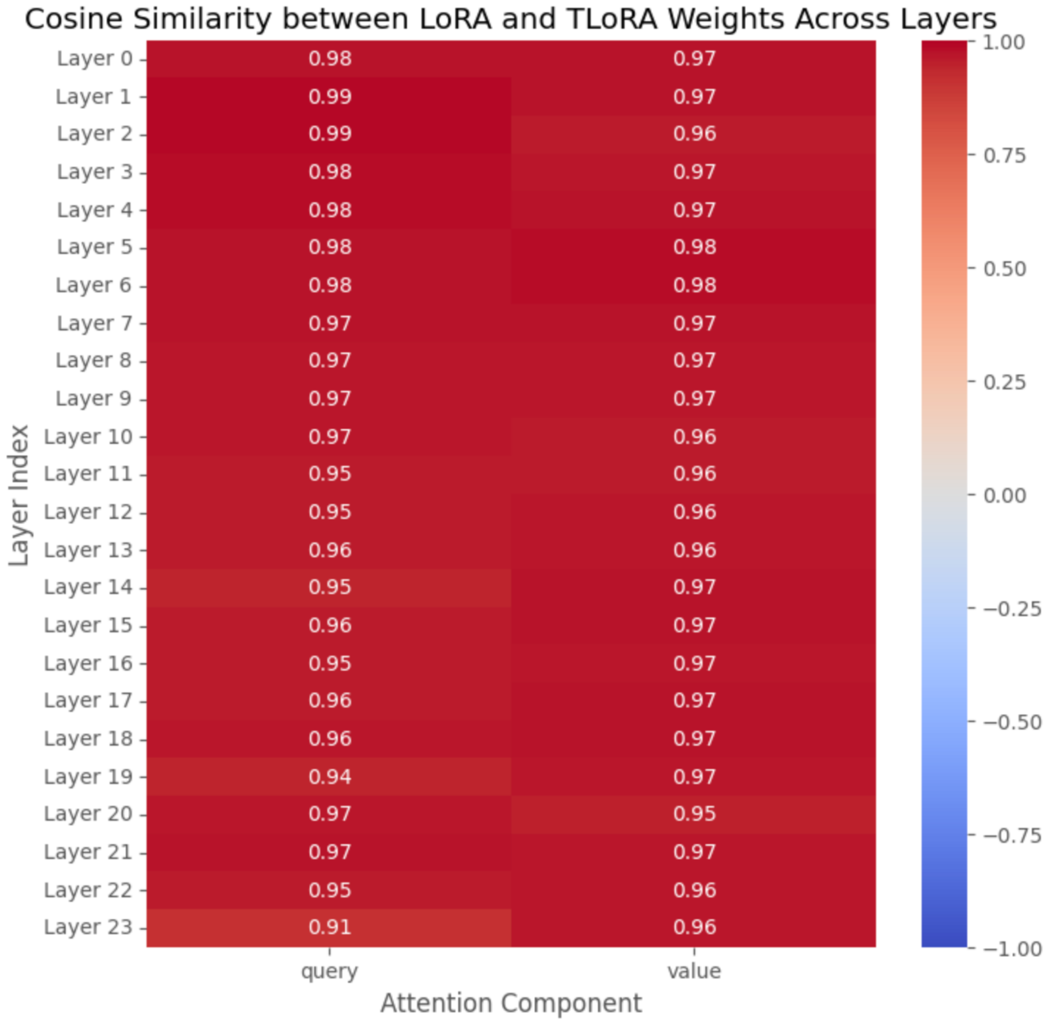

Figure 10: Cosine similarity values across all layers for the query (q) and value (v) attention components between TLoRA and LoRA.

To further quantify the similarity between LoRA and TLoRA, we calculate the cosine similarity between their learned parameter updates across all layers for both the query (q) and value (v) attention components. As shown in Figure 10, the cosine similarity values are consistently close to 1 across layers, indicating a high degree of alignment in the adaptation directions achieved by

TLoRA compared to LoRA. This high similarity suggests that, despite TLoRA's unique tri-matrix decomposition with two fixed matrices and a learnable scaling factor, it effectively approximates the directional adjustments made by LoRA. The close alignment for both (q) and (v) components indicates that TLoRA captures the essential transformations required for task-specific adaptation in a manner nearly indistinguishable from LoRA.

These findings provide strong evidence that TLoRA replicates LoRA's core adaptation behavior, achieving comparable fine-tuning efficiency and effectiveness. By capturing the same key parameter shifts, TLoRA demonstrates that it is capable of maintaining LoRA-like performance even with its modified, parameter-efficient architecture. This alignment highlights TLoRA's potential as a resource-efficient fine-tuning strategy that can emulate the effectiveness of traditional low-rank adaptations.

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, C to compute a low-rank update to the model's weights. The contribution of this update is controlled by a trainable scaling factor α, 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.

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

Brown, Tom B. 2020. "Language Models Are Few-Shot Learners." *arXiv Preprint arXiv:2005.14165*.

Chen, Zezhou, Zhaoxiang Liu, Kai Wang, and Shiguo Lian. 2024. "Reparameterization-Based Parameter-Efficient Fine-Tuning Methods for Large Language Models: A Systematic Survey." In *CCF International Conference on Natural Language Processing and Chinese Computing*, 107–18. Springer.

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.

Floridi, Luciano, and Massimo Chiriatti. 2020. "GPT-3: Its Nature, Scope, Limits, and Consequences." *Minds and Machines* 30:681–94.

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*.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet Classification." In *Proceedings of the IEEE International Conference on Computer Vision*, 1026–34.

Hu, Edward J, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. "Lora: Low-Rank Adaptation of Large Language Models." *arXiv Preprint arXiv:2106.09685*.

Kenton, Jacob Devlin Ming-Wei Chang, and Lee Kristina Toutanova. 2019. "Bert: Pre-Training of

Deep Bidirectional Transformers for Language Understanding.” In *Proceedings of naacL-HLT*, 1:2. Minneapolis, Minnesota.

Kopiczko, Dawid J, Tijmen Blankevoort, and Yuki M Asano. 2023. “Vera: Vector-Based Random Matrix Adaptation.” *arXiv Preprint arXiv:2310.11454*.

Li, Yang, Shaobo Han, and Shihao Ji. 2024. “VB-LoRA: Extreme Parameter Efficient Fine-Tuning with Vector Banks.” *arXiv Preprint arXiv:2405.15179*.

Lialin, Vladislav, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. 2023. “Relora: High-Rank Training through Low-Rank Updates.” In *The Twelfth International Conference on Learning Representations*.

Liu, Haokun, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. “Few-Shot Parameter-Efficient Fine-Tuning Is Better and Cheaper than in-Context Learning.” *Advances in Neural Information Processing Systems* 35:1950–65.

Liu, Yinhan. 2019. “Roberta: A Robustly Optimized Bert Pretraining Approach.” *arXiv Preprint arXiv:1907.11692* 364.

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*.

Matena, Michael S, and Colin A Raffel. 2022. “Merging Models with Fisher-Weighted Averaging.” *Advances in Neural Information Processing Systems* 35:17703–16.

Nangia, Nikita, and Samuel R Bowman. 2019. “Human vs. Muppet: A Conservative Estimate of Human Performance on the GLUE Benchmark.” *arXiv Preprint arXiv:1905.10425*.

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*.

Ouyang, Long, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, et al. 2022. “Training Language Models to Follow Instructions with Human Feedback.” *Advances in Neural Information Processing Systems* 35:27730–44.

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.

Touvron, Hugo, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, et al. 2023a. “Llama: Open and Efficient Foundation Language Models.” *arXiv Preprint arXiv:2302.13971*.

———. 2023b. “Llama: Open and Efficient Foundation Language Models.” *arXiv Preprint arXiv:2302.13971*.

Wang, Alex. 2018. “Glue: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding.” *arXiv Preprint arXiv:1804.07461*.

Wang, Xi, Laurence Aitchison, and Maja Rudolph. 2023. “LoRA Ensembles for Large Language Model Fine-Tuning.” *arXiv Preprint arXiv:2310.00035*.

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)*, 2786–92. IEEE.

Xu, Lingling, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. 2023. “Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment.” *arXiv Preprint arXiv:2312.12148*.

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*.