Tensor Train Low-rank Approximation (TT-LoRA): Democratizing AI with Accelerated LLMs

Afia Anjum*[‡], Maksim E. Eren[†], Ismael Boureima*, Boian Alexandrov*, Manish Bhattarai*,
*Theoretical Division, Los Alamos National Laboratory. Los Alamos, USA.

[†]Advanced Research in Cyber Systems, Los Alamos National Laboratory. Los Alamos, USA.

[‡]University of Texas at Arlington. Texas, USA.

Abstract—In recent years, Large Language Models (LLMs) have demonstrated remarkable capabilities across a wide range of natural language processing (NLP) tasks, such as questionanswering, sentiment analysis, text summarization, and machine translation. However, the ever-growing complexity of LLMs demands immense computational resources, hindering the broader research and application of these models. To address this, various parameter-efficient fine-tuning strategies, such as Low-Rank Approximation (LoRA) and Adapters, have been developed. Despite their potential, these methods often face limitations in compressibility. Specifically, LoRA struggles to scale effectively with the increasing number of trainable parameters in modern large scale LLMs. Additionally, Low-Rank Economic Tensor-Train Adaptation (LoRETTA), which utilizes tensor train decomposition, has not yet achieved the level of compression necessary for fine-tuning very large scale models with limited resources. This paper introduces Tensor Train Low-Rank Approximation (TT-LoRA), a novel parameterefficient fine-tuning (PEFT) approach that extends LoRETTA with optimized tensor train (TT) decomposition integration. By eliminating Adapters and traditional LoRA-based structures, TT-LoRA achieves greater model compression without compromising downstream task performance, along with reduced inference latency and computational overhead. We conduct an exhaustive parameter search to establish benchmarks that highlight the trade-off between model compression and performance. Our results demonstrate significant compression of LLMs while maintaining comparable performance to larger models, facilitating their deployment on resource-constraint platforms.

Index Terms—Tensor-train, Low Rank Approximation, Large Language Model, BERT, Compression

I. INTRODUCTION

Large Language Models (LLMs) such as LLaMA-70B [1], ChatGPT-4 [2], Bard [3], and Claude [3] represent a significant step towards Artificial General Intelligence (AGI) [4]. These models are trained using vast amounts of data and are complex neural network architectures, such as Transformers [5], enabling the models to excel in interpreting complex linguistic patterns. Consequently, these models can accurately recognize, translate, predict, and generate text, along with performing other content-related tasks. The effectiveness of LLMs in capturing the nuances of language has made them indispensable tools for driving previously unattainable innovations.

While pre-trained LLMs provide a robust foundation for general language tasks, fine-tuning these models on

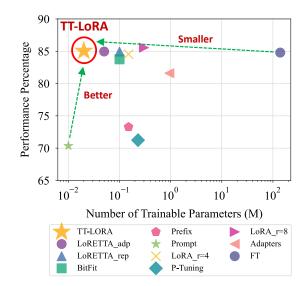


Fig. 1: Performance vs. trainable parameters comparison between TT-LoRA and various PEFT methods on DeBERTa model using the GLUE benchmark

application-specific datasets is crucial for optimizing performance in specialized applications, which allows the models to adapt to a particular context or domain. However, fine-tuning involves adapting all the parameters of a pre-trained model to a new tasks, which poses significant challenges due to the reliance of these models on increasingly complex Transformers with exploding parameter counts (model size in billions of parameters), requiring substantial computational resources [6]. Fine-tuning these large models through traditional approaches, which involves scaling these models using multiple GPUs, is hindered by resource limitations and rising costs, monopolizing research access and raising ethical concerns. Additionally, the immense computational demands of these models raise serious environmental concerns due to the energy consumption of massive computing complexes [7].

Since full model fine-tuning, which involves adjusting all the parameters of a pre-trained model, becomes prohibitively expensive as the model size of LLMs grows, notable pragmatic efforts, such as parameter-efficient fine tuning (PEFT) techniques like Adapters [8], Prefix Tuning [9], Prompt Tuning [10], Low-Rank Adaptation (LoRA) [11], and Low-Rank Economic Tensor-Train Adaptation (LoRETTA) [12]

have been proposed for efficient LLM fine-tuning. However, Adapter-based PEFT methods increase model's inference latency while Prompt and Prefix Tuning sacrifice model accuracy. Moreover, when scaling to recently proposed larger models such as LAaMA3-70B and Mixture of Experts (MoE), the number of trainable parameters needed in LoRA and LoRETTA methods is still high, limiting these models' scalability and application.

To address the scalability issues of PEFT methods with increasing model parameters, in this paper, we propose Tensor Train Low Rank Approximation (TT-LoRA), a novel PEFT approach using Tensor Train (TT) decomposition [13]. TT decomposition has already shown promise in compressing and accelerating neural networks by efficiently representing large weight matrices (model parameters) in a compact tensor format, facilitating substantial reductions in computational load without severely compromising performance [14], [15]. In addition, the LoRETTA approach has demonstrated the efficacy of utilizing tensor train (TT) decomposition for refining weight updates, achieving notable accuracy enhancements in LLM applications. Motivated by the efficiency of TT decomposition, we explore its potential and extend LoRETTA with architectural variations. Our approach, TT-LoRA, diverges significantly by optimizing the integration process of TT decomposition into the model's architecture, specifically, omitting Adapters and the LoRA-based structure employed in LoRETTA. This crucial modification eliminates the additional inference latency associated with these elements, achieves superior model compression, and reduces model complexity and computation overhead.

Our contributions are:

- We introduce Tensor Train Low Rank Approximation (TT-LoRA), a parameter-efficient fine-tuning strategy for large language models (LLMs). TT-LoRA leverages tensor train decomposition to enable fine-tuning of LLMs while significantly reducing the number of trainable parameters.
- We conduct a comprehensive evaluation of TT-LoRA's performance across a variety of downstream tasks and LLMs of differing scales, including BERT-based models and the larger-scale LLaMA-2 and LLaMA-3 models.
- We benchmark the performance of TT-LoRA against other widely PEFT methods across diverse model scales.
 Our results show that TT-LoRA achieves greater performance while ensuring significant model compression, as shown in Figure 1.
- We perform an exhaustive parameter search to establish benchmarks that illustrate the trade-offs between model compression and performance, providing valuable insights for optimizing PEFT methods.

II. RELATED WORKS

In this section, we explore various strategies for parameterefficient fine-tuning of LLMs, as illustrated in Figure 2, which are crucial for deploying these models in resource-constrained environments. Adapter-based Methods: Adapter-based methods [16]–[18] introduce small, trainable modules within the pre-trained model's architecture, where each module consists of fully connected layers configured with a bottleneck structure (Figure 2a). keeping most of the model's parameters unchanged. This adapter-based design keeps the pre-trained model's weights fixed while only updating the parameters within the adapters during task-specific fine-tuning. Although adapter-based methods can reduce the number of trainable parameters, they introduce additional computational steps in the Transformer blocks. Due to their sequential processing nature, these adapter layers do not effectively leverage hardware parallelism, which results in increased latency, particularly in online inference scenarios with small batch sizes [19].

Prefix Tuning, Prompt Tuning and P-tuning: Li and Liang [9] introduced prefix-tuning, a method where a sequence of continuous task-specific vectors, also known as prefix, is prepended to the model input, optimizing only the prefix while keeping the model parameters frozen. Lester et al. [10] simplify the prefix-tuning by proposing prompt-tuning, where k tunable tokens per downstream tasks are prepended to the input text. Liu et al. [20] proposed p-tuning, which extends prompt tuning by integrating continuous token embeddings not only at the input level but also at various points throughout the model, enhancing the adjustment of model processing. These tuning methods are illustrated in Figure 2b. However, these tuning methods occupy a part of the fixed sequence length that transformer-based architectures can process. Consequently, this reduces the available space for actual task-related input, potentially compromising the model's efficiency [19].

Low-rank Approximation: Hu et al. [19] proposed a Low-rank Adaptation (LoRA) fine-tuning approach leveraging matrix factorization, which decomposes a large matrix into a product of two or more smaller matrices (Figure 2c). For a pre-trained weight matrix $W_0 \in \mathbb{R}^{m \times n}$, the weight update in a full fine-tuning setting would be $W_0 + \Delta W$, where $\Delta W \in \mathbb{R}^{m \times n}$. In contrast, LoRA enables low-rank updates to the weight matrix as $W_0 + BA$, where W_0 is kept frozen while only optimizing B and A matrices. Here, BA is the lowrank approximation of ΔW , where $B \in \mathbb{R}^{m \times r}$, $A \in \mathbb{R}^{r \times n}$, and $r \ll min(m, n)$. While LoRA achieves similar or even better performance than full-model fine-tuning, it still incurs a large number of trainable parameters. For instance, when finetuning the LLaMA-2-70B model using LoRA, over 16 million parameters need to be updated, exceeding the total number of parameters in some BERT models [12].

Tensor-based Model Compression: Yang et al. [12] proposed Low-Rank Economic Tensor-Train Adaptation (LoRETTA), inspired by the Tensor Train (TT) format initially explored by Novikov et al. [21], which represents a matrix with a series of tensor factors. The authors proposed two methods, LoRETTA $_{adp}$ and LoRETTA $_{rep}$. The former method, LoRETTA $_{adp}$, employs tensorized adapters, compressing the weight updating matrix using two tensorized linear layers (Figure 2d). The latter method, LoRETTA $_{rep}$, performs matrix factorization to reduce the large updating matrix into two small

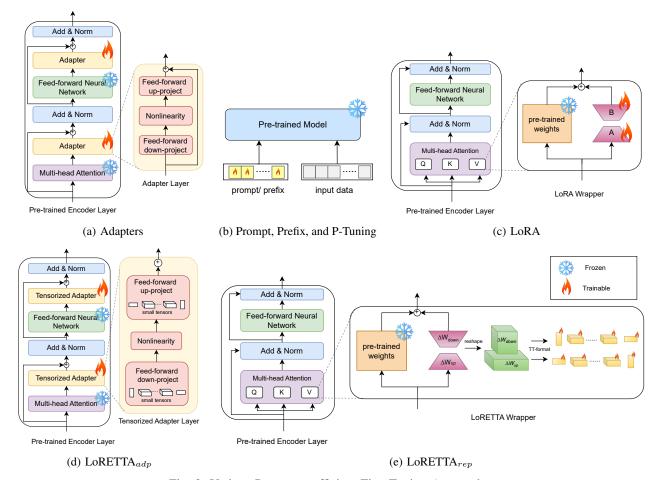


Fig. 2: Various Parameter-efficient Fine-Tuning Approaches

matrices, followed by reshaping the two updating matrices into small tensor factors (Figure 2e). However, when scaling to recent larger models such as LLaMA2-70B or attempting to leverage the full potential of techniques like Mixture of Experts (MoE), which are inherently resource-intensive, LoRETTA still incurs a large number of trainable parameters.

Our approach, TT-LoRA, employs a comprehensive parameter search to optimize weight updating matrix decomposition. This process involves identifying the optimal tensor shapes and hyperparameters to achieve the most efficient compressed representation with higher accuracy. Unlike LoRETTA, which employs Adapters and the LoRA-format, TT-LoRA directly decomposes the weight updating matrix into small tensors. This approach eliminates the potential for increased inference latency associated with Adapters and achieves more effective compression by avoiding the LoRA-based structure. Extensive evaluations have demonstrated that TT-LoRA surpasses LoRETTA's performance on BERT and LLaMA, both on accuracy and model compression, across various classification tasks, establishing it as the superior method for efficient and accurate model representation.

III. TENSOR TRAIN BASED LOW-RANK ADAPTATION (TT-LORA)

In this section, we first formulate the objective function of fine-tuning an LLM and outline the challenges associated with conventional full fine-tuning approach. Subsequently, we introduce our proposed PEFT method TT-LoRA, detailing its design and how it addresses these challenges.

A. Problem Statement

Let $P_{\theta}(y, x)$ be a pre-trained language model, parameterized by θ . For example, $P_{\theta}(y, x)$ could represent a versatile multitask learning model, such as DistilBERT [22], DeBERTa [23], or LLaMA-3 [1], all of which are developed from the Transformer architecture [24] initially proposed by Vaswani et al. in 2017. We consider adapting the pre-trained model to downstream tasks, such as text classification, summarizing, question answering, and sentiment analysis. Each downstream task is represented by a training dataset of input-output pairs: $\mathcal{D}: (x_i, y_j)_{i=1}^N, y_{j=1}^M$. Here, x_i represents a sequence of input tokens. The output y_i can vary depending on the task, for instance, it may be a sequence of tokens for text generation tasks, categorical labels for classification tasks, or continuous values for regression tasks. For example, for sentiment analysis, x_i is a social media post, and y_i is the categorical label; for the question-answering task, x_i is the question, and

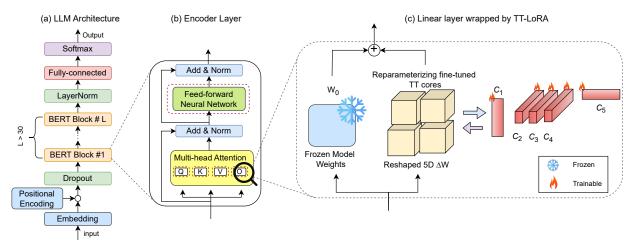


Fig. 3: Overview of the proposed TT-Lora framework: (a) BERT Architecture, (b) Encoder layer of BERT Architecture, (c) TT-LoRA wrapping the *value* vector of BERT encoder: the pre-trained weight, W_0 , is kept frozen while the update to the pre-trained weight, ΔW , is decomposed into small tensors.

 y_j is the answer; for the summarizing task, x_i is the article and y_j is the summary of the corresponding article. Consider $P_{\theta}(y,x)$ to be a text generation task-based pre-trained model. During the full fine-tuning, the model is initialized with its pre-trained weights θ_0 and subsequently updated to $\theta_0 + \Delta \theta$. Here, $\Delta \theta$ represents the modifications made to the pre-trained weights, specifically tailored to enhance performance on the downstream task. These adjustments are derived by optimizing the following objective function: [19]:

$$\max_{\theta} \sum_{(x,y)\in\mathcal{D}} \sum_{t=1}^{|y|} \log(P_{\Phi}(y_t \mid x, y_{< t})) \tag{1}$$

Here, the objective function aims to maximize the cumulative log probability of correctly predicting each output token y_t , given the input sequence x and all preceding output tokens.

A major drawback of full fine-tuning is that all the model parameters need to be trained for each downstream task, which makes the dimension of $|\Delta\theta|$ equal to $|\theta_0|$. In this paper, we introduce a parameter-efficient fine-tuning strategy that significantly decreases the number of trainable parameters from θ_0 to ϕ , where the dimension of $|\Delta\phi| \ll |\Delta\theta|$.

B. Proposed Method

Transformer-based embedding models, such as BERT, depicted in Figure 3(a), comprise multiple dense layers, including numerous encoder blocks and a fully connected feed-forward neural network. Each encoder block, detailed in Figure 3(b), contains a complex sub-layer arrangement, specifically multi-head attention layer and a fully connected feed-forward neural network. The weight matrices associated with these dense layers possess full rank, ensuring that each input feature uniquely contributes to the output and maximizes the learning capacity of the layer. However, when adapting to a specific downstream task, Aghajanyan et al. [25] show that the pre-trained LLMs have a low intrinsic dimension and Hu et al. [19] hypothesize that the updates to the weights of the pre-trained models also have a low intrinsic rank, highlighting

a potential area for model compression during adaptation to downstream tasks. Therefore, in this paper, for a pre-trained weight matrix $W_0 \in \mathbb{R}^{m \times n}$ and its corresponding update during adaptation $\Delta W \in \mathbb{R}^{m \times n}$, we constrain the update through the proposed TT-LoRA approach by presenting ΔW with low-rank representations, as shown in Figure 3(c).

To achieve a low-rank approximation of ΔW , we utilize tensor decomposition [26], more specifically, Tensor Train (TT) [27] decomposition. TT decomposition decomposes a tensor into a series of low-rank, small, three-dimensional tensors (cores). The product of these low-rank tensor cores provides an accurate approximation of the original tensor, significantly reducing its dimensionality while preserving essential structure and information. The fundamental property of TT decomposition is that each core tensor interacts only with its immediate predecessor and successor. This localized interaction allows operations, such as tensor core multiplication to approximate the original tensor, to be performed in a step-by-step manner, focusing on smaller, manageable pieces rather than the entire tensor at once. As a result, the complexity of tensor operations is significantly simplified, leading to substantial reductions in computational and memory requirements [13]. However, TT decomposition is particularly applicable to high-dimensional tensors. Therefore, in TT-LoRA, the matrix ΔW , which is initialized using a random Gaussian distribution, is first represented as a d-dimensional tensor $\Delta W \in \mathbb{R}^{k_1 \times \cdots \times k_d}$. Here, $\prod_{i=1}^{d} k_i = m \times n$. The d-dimensional tensor ΔW is then decomposed into d number of small tensor cores $\mathcal{C}_1,\ldots,\mathcal{C}_d$. The shape of each tensor core can be defined as $C_i \in \mathbb{R}^{r_{i-1}, k_i, r_i}$, given the TT rank $[r_0, \dots, r_d]$, where the first (r_0) and last (r_d) TT ranks are 1. Consequently, the total number of parameters in the tensor train decomposition of ΔW can be represented as:

$$\Delta \mathcal{W} \approx \bigcup_{i=1}^{d} C_i \in \mathbb{R}^{\sum_{i=1}^{d} r_{i-1} \times k_i \times r_i}$$
 (2)

Here, $r_{i-1} \times k_i \times r_i$ is the size of the *i*-th tensor core. During

fine-tuning, the pre-trained weight matrix W_0 remains frozen and does not receive gradient updates, while the $\Delta \mathcal{W}$ contains trainable parameters. The adapted weight matrix $W_{adapted}$ is given by:

$$W_{adapted} = W_0 + \alpha(\Delta W) \tag{3}$$

Using TT decomposition:

$$W_{adapted} = W_0 + \alpha \left(\prod_{i=1}^{d} C_i\right) \tag{4}$$

Thus, this adapted layer applies a linear transformation to an input x can be described as follows:

$$y = W_0 x + \alpha \left(\prod_{i=1}^{d} C_i x \right) \tag{5}$$

Here, W_0 represents the pre-trained weight matrix of shape $m \times n$, ΔW is the update to the weight matrix in full fine-tuning of shape $m \times n$, ΔW is the d-dimensional tensorized matrix of shape $k_1 \times \cdots \times k_d$, C_i is the *i*-th tensor core of shape $r_{i-1} \times k_i \times r_i$ and α is a fixed scaling factor used to scale the update before adding to the pre-trained weight.

The compression ratio of TT-LoRA is closely related to the choice of TT ranks and the structure of d-dimensional tensor $\Delta \mathcal{W}$. For instance, consider a W_0 with dimensions 768×2304 . In full fine-tuning, ΔW will have the exact dimensions of 768×2304 , resulting in 1,769,472 trainable parameters. However, with TT-LoRA, considering $\Delta \mathcal{W}$ as 7D tensor of shape $12 \times 8 \times 8 \times 3 \times 8 \times 8 \times 12$ with TT-rank of 5 results in 1,135 trainable parameters, achieving a remarkable 1560x compression ratio. Reducing the TT-rank and/or increasing the tensor dimension increases the compression ratio even further. To obtain the optimal TT-rank and tensor dimensions for our proposed method, we did a thorough hypermarameter search which is discussed in Section IV-D.

IV. RESULTS AND DISCUSSION

We conduct experiments to evaluate the performance of TT-LoRA on various downstream tasks, ranging from natural language understanding (NLU) to generation (NLG), utilizing pre-trained LLMs of different scales. Specifically, we evaluate DeBERTa [23] and RoBERTa [28] on Generalized Language Understanding Evaluation (GLUE) [29] benchmark while utilizing SuperGLUE [30] benchmark on larger-scale models, such as LLaMA-2-7B [31] and LLaMA-3-8B [1]. In addition to reporting model performance on downstream tasks with TT-LoRA, we compare the results with baseline fine-tuning methods, such as Full fine-tuning (FT), Adapters [32], Prompt tuning [10], Prefix Tuning [9], P-tuning [20], BitFit [33], LoRA [19], and LoRETTA [12]. Furthermore, we conduct an extensive search for optimal parameters, including the best tensor shapes for ΔW and the appropriate TT ranks. This effort aims to establish benchmarks that illustrate the tradeoff between model compression and performance.

For this paper, our experiments utilized a system that integrates four NVIDIA Hopper (H100) GPUs, each paired

with a corresponding NVIDIA Grace CPU via NVLink-C2C, facilitating rapid data transfer crucial for intensive computational tasks. The GPUs are equipped with 96GB of HBM2 memory, optimal for handling large models and datasets.

A. GLUE Experiments on BERT Family

We initially conducted experiments on (RoBERTa) Robustly Optimized BERT Pretraining Approach [28], which is an optimized method for training BERT (Bidirectional Encoder Representations from Transformers) [34], a transformer-based LLM. Developed by researchers at Meta AI, RoBERTa revises BERT's pretraining methodology to improve the model's performance in several ways, such as dynamic masking, eliminating the next sentence prediction loss, and increasing the batch size while decreasing the learning rate. Apart from RoBERTa, We performed experiments on DeBERTa (Decoding-enhanced BERT with disentangled Attention) [23], a recent variant of BERT trained on a larger scale. Developed by Microsoft, DeBERTa improves the BERT architecture by introducing a novel disentangled attention mechanism that separately models the content and position, enhancing the model's ability to understand contextual relationships in text. We utilized the pre-trained RoBERTa-base and DeBERTa-base from the HuggingFace Transformers library.

In RoBERTa and DeBERTa architecture, each encoder layer includes four weight matrices within the self-attention module (W_q, W_k, W_v, W_o) and a feed-forward neural network (FFNN). While TT-LoRA can be applied to any of the weight matrices in a neural network to reduce the number of trainable parameters, our experiments specifically target W_q and W_v . This focus aligns with findings from the LoRA paper, which indicates that models achieve optimal performance when these particular weights are fine-tuned, while the remaining weights are kept frozen [19]. We performed hyperparameter optimization by fine-tuning the model with different TT-LoRA parameter initializations, the details of which will be presented later. Using the HyperBand optimizer [35], we identified the most efficient parameters. For each run, the model was finetuned for up to 20 epochs with an early stopping criterion of 5 epochs. Specifically, the training was halted if the validation loss did not improve for 5 consecutive epochs. The best model was selected based on the lowest observed validation loss from these runs. For reporting performance on the GLUE benchmark tasks, we use the following metrics: matched accuracy for MNLI, Matthews correlation coefficient for CoLA, Spearman correlation coefficient for STS-B, F1 score for both MRPC and QQP, and accuracy for all other tasks. Table I summarizes the downstream task performance comparison between TT-LoRA and other baseline PEFT methods.

As shown in Table I, TT-LoRA consistently achieves superior or comparable performance to other PEFT methods when FT DeBERTa on GLUE tasks, with no more than 0.2M trainable parameters. Consequently, TT-LoRA stands out for its efficiency by outperforming 9 out of 10 FT approaches in both model compression and accuracy. Prompt Tuning, which utilizes just 0.01M trainable parameters compared to TT-

TABLE I: Comparative analysis of various PEFT methods on the BERT family models

Model & Method	# Train. Param.	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
DeBERTa-Base (FT)*	139.19M	88.67	94.61	91.98	59.32	93.04	91.42	68.23	91.10	84.79
DeBERTa-Base (Adapters $_{r=8}$)*	0.94M	87.69	94.72	88.88	54.19	92.95	85.52	59.20	89.68	81.60
DeBERTa-Base (LoRA $_{r=8}$)*	0.30M	87.30	94.95	92.84	60.56	93.35	85.19	80.14	90.13	85.56
DeBERTa-Base (P-Tuning)*	0.23M	56.25	91.39	79.93	43.31	86.30	78.43	55.95	78.38	71.24
DeBERTa-Base (LoRA $_{r=4}$)*	0.15M	87.69	94.49	91.10	62.57	92.60	87.30	69.67	91.12	84.54
DeBERTa-Base (Prefix)*	0.15M	60.32	88.87	81.22	45.82	83.28	82.22	59.57	84.99	73.28
DeBERTa-Base (BitFit)*	0.10M	84.63	95.41	91.42	64.06	93.30	84.15	66.79	90.23	83.75
DeBERTa-Base (LoRETTA _{adp})*	0.10M	85.93	95.30	93.53	60.84	92.99	84.08	75.50	91.32	84.96
DeBERTa-Base (LoRETTA _{rep})*	0.05M	86.80	95.53	88.73	59.69	93.25	89.2	75.81	90.66	84.95
DeBERTa-Base (Prompt)*	0.01M	77.63	92.43	81.90	32.99	80.30	78.15	62.81	56.71	70.36
DeBERTa-Base (TT-LoRA)	0.02M	83.1	94.15	90.68	70.26	91.01	85.57	75.09	90.57	85.05
RoBERTa-Base (FT)*	124M	86.27	93.46	88.97	74.20	91.49	91.20	77.61	90.04	86.66
RoBERTa-Base (LoRA $_{r=8}$)*	0.63M	86.82	94.01	91.48	62.08	92.39	85.71	74.51	90.48	84.69
RoBERTa-Base (BitFit)*	0.10M	85.30	94.80	92.33	62.70	91.30	68.10	73.60	88.50	82.08
RoBERTa-Base (LoRETTA _{adp})*	0.10M	85.61	94.38	91.08	62.70	92.12	87.22	78.70	90.26	85.26
RoBERTa-Base (TT-LoRA)	0.02M	85.76	93.57	87.05	72.82	91.23	88.06	77.25	91.23	86.00

Note: * represents results shown in previous work [12].

TABLE II: Comparative analysis of various PEFT methods on the LLaMA family models

Model &	LLaMA2-7B					LLaMA3-8B			
Task	FT*	Adapter*	$LoRA_{r=8}*$	Prefix*	$Loretta_{rep}^*$	LoRETTA _{adp} *	TT-LoRA	$LoRA_{r=8}$	TT-LoRA
CB	66.07	66.07	67.86	51.78	55.35	66.07	85.71	71.43	85.71
BoolQ	84.6	71.8	84.8	78.6	78.1	87.0	86.78	88.93	88.13
WSC	63.46	62.50	62.50	61.53	57.61	63.46	67.30	63.46	66.35
COPA	86	84	81	83	86	87	81	72	77.99
Avg.	75.03	71.09	74.04	68.72	69.26	75.88	80.19	73.96	79.55
#Train. Param.	6738.42M	50.33M	4.19M	1.31M	0.51M	0.88M	0.1M	3.41M	0.2M

Note: * represents results shown in previous work [12].

LoRA's 0.02M, surpasses TT-LoRA in model compression but significantly suffers in performance. Nevertheless, TT-LoRA requires merely twice the trainable parameters of Prompt Tuning while achieving, on average, about 15% higher model accuracy. TT-LoRA also achieves achieves substantial model compression when FT RoBERTa, significantly reducing the number of trainable parameters. Specifically, TT-LoRA has reduced trainable parameters by approximately factors of $6,200\times$ for full FT, $31.5\times$ for LoRA, $5\times$ for BitFit, and $5\times$ for LoRETTA $_{adp}$. Despite this reduction, TT-LoRA outperforms other PEFT methods, except for full FT, in average model accuracy across various GLUE task sets.

B. SuperGLUE Experiments on LLaMA Family

Encouraged by the outcomes observed with DeBERTa and RoBERTa models, we extended our experiments to include the larger-scale Large Language Model at Meta AI (LLaMA) [36] models. LLaMA is a series of larger-scale language models designed for various natural language understanding and generation tasks. In our experiments, we utilized LLaMA-2-7B and LLaMA-3-8B models to assess the effectiveness of our proposed PEFT method. We chose these LLaMA models due to their extensive parameter sets, ranging into the billions, which provide a rigorous test environment to evaluate how well our PEFT approach can compress and optimize these substantial models while maintaining comparable performance levels. For our experiments, we utilized the pre-trained LLaMA2-7b and LLaMA3-8b models available in HuggingFace Transformers library. We conducted a comparative analysis of TT-LoRA against other baseline PEFT methods using the SuperGLUE benchmark tasks and the results are summarized in Table II. Similar to BERT family experiments, TT-LoRA has been applied to W_q and W_v weight matrices of the self-attention module of LLaMA models and has been trained over multiple epochs, stopping the process if the validation loss did not improve for 5 consecutive epochs. The best model is then chosen from these runs based on the lowest observed validation loss. For reporting performance on the SuperGLUE benchmark tasks, we used F1 score for both CB and WSC tasks, and accuracy for both BoolQ and COPA tasks.

Table II highlights TT-LoRA's performance on the LLaMA2-7B model, where it consistently outperforms all other PEFT methods on CB and WSC tasks and all but LoRETTA $_{rep}$ on the BoolQ task. It achieves comparable performance on the COPA task and, on average, surpasses all competing PEFT methods. This higher model performance is achieved while attaining significant reductions in trainable parameters compared to our baselines by factors of approximately 67,384x (full fine-tuning, FT), 500x (Adapter), 41.9x (LoRA $_{r=8}$), 13.1x (Prefix), 5.1x (LoRETTA $_{rep}$), and 8.8x (LoRETTA $_{adp}$).

We extended our experiment by FT the LLaMA3-8B model on the SuperGLUE benchmark and compared its performance against LoRA. As shown in Table II, TT-LoRA either matched or exceeded the performance of LoRA across all tasks while achieving a remarkable reduction in trainable parameters, which is approximately $170.5 \times$ fewer parameters compared to LoRA.

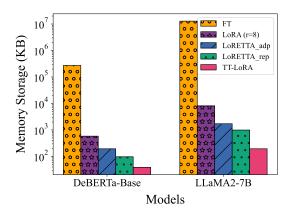


Fig. 4: Memory storage comparison for trainable parameters of various PEFT methods and LLMs.

C. Memory Performance

We evaluate the storage requirements of various PEFT methods applied to fine-tune the DeBERTa and LLaMA2-7B models. Specifically, we select the three top-performing PEFT methods from Table I and Table II. The storage calculations are based on the assumption that the model weights are stored with 16-bit precision. As demonstrated in Figure 4, TT-LoRA reduces the storage requirements for trainable parameters when used with DeBERTa, necessitating only 39 KB. This represents a significant reduction in storage needs, achieving reductions by factors of approximately $2.5\times$, $5\times$, $15\times$, and $7000 \times$ compared to LoRETTA_{rep}, LoRETTA_{adp}, LoRA_{r=8}, and full FT, respectively. A similar memory efficiency is achieved through TT-LoRA when used with LLaMA2-7B, requiring approximately 195 KB of storage for trainable parameters. This storage efficiency significantly surpasses that of other PEFT methods, compared to our baselines reducing storage demands by factors of roughly $5 \times (LoRETTA_{rep})$, $8.8 \times \text{(LoRETTA}_{adp}), 42 \times \text{(LoRA}_{r=8}), \text{ and } 65,925 \times \text{(full)}$ fine-tuning, FT). The reduced storage requirement of TT-LoRA establishes it as a highly efficient approach for FT LLMs, particularly on hardware with limited resources. This efficiency facilitates broader deployment options and ensures that larger-scale LLM capabilities are accessible even in constrained environments.

D. Hyperparameters

To optimize the performance of TT-LoRA, we conducted a comprehensive hyperparameter search using Ray Tune [37], a scalable hyperparameter optimization framework. The hyperparameters are selected to ensure a robust comparison of model performance across different configurations and tasks. We employed the HyperBand optimizer [35] to explore a discrete search space for hyperparameter combinations, aiming to minimize validation loss with fewer simulations. This approach allowed us to estimate the optimal parameters efficiently, performing around 200 searches. The search space included learning rates and specific model-related parameters

TABLE III: Hyperparameter Search Used for Experiments

TABLE III. Hyperparameter Search Osed for Experimen							
		[64, 36, 12, 64]					
	DeBERTa	[32, 12, 3, 4, 12, 32]					
		[12, 8, 8, 3, 8 , 8, 12]					
		[32, 16, 2, 3, 3, 3, 2, 32]					
		[32, 4, 2, 2, 2, 3, 3, 3, 2, 32]					
		[16, 2, 4, 2, 2, 2, 3, 3, 3, 2, 2, 16]					
	RoBERTa	[64, 16, 9, 64]					
		[12, 8, 8, 8, 8, 12]					
Tensor Shape		[12, 8, 8, 2, 4, 8, 12]					
1		[12, 8, 8, 2, 2, 2, 8, 12]					
		[8, 6, 2, 2, 4, 4, 2, 2, 6, 8]					
		[8, 6, 2, 2, 2, 2, 2, 2, 2, 6, 8] [128, 32, 32, 128]					
		[16, 16, 16, 16, 16]					
	LLaMA2-7B						
	LLaMA3-8B	[16, 16, 4, 4, 4, 4, 16, 16]					
	22411111002	[16, 8, 4, 4, 2, 2, 4, 4, 8, 16]					
		[16, 4, 4, 4, 2, 2, 2, 2, 4, 4, 4, 16]					
	DeBERTa	5, 8, 10, 12, 16					
Ranks	RoBERTa						
Kaliks	LLaMA2-7B	1, 2, 4, 5, 8, 10, 12, 16					
	LLaMA3-8B						
Alpha		, 2, 4, 8, 10, 12, 16, 32					
Learning Rate	1e -	5, 1e-4, 5e-5, 5e-4					
Model							
○ TT-LoR	A	LoRA > Adapters					
☆ LoRET	TA rep	Prefix < FT					
	TA_adp						

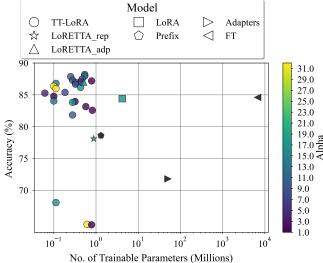


Fig. 5: Model compression vs. accuracy of LLaMA2-7B model with TT-LoRA having different tensor format and other PEFT approaches on SuperGLUE BoolQ benchmark.

such as tensor shapes, ranks, and alpha values, as highlighted in Table III. The best model configurations were determined based on the lowest validation loss observed during the tuning Ray Tune trials. The optimal hyperparameters for TT-LoRA, such as tensor shape, tensor rank, α , and learning rate are summarized in Table IV. The hyperparameters used for other PEFT methods are demonstrated in [12].

In addition to identifying the optimal model configuration, the logs generated by Ray Tune provide a detailed analysis of the relationship between model performance and the number of trainable parameters. This insight is crucial for optimizing a PEFT method, especially in resource-constrained environments where efficiency is also important. Figure 5 illustrates how the number of trainable parameters, determined by specific tensor shapes and ranks, affects the accuracy of

TABLE IV: Optimal Hyperparameters for TT-LoRA

Model	IIvim amm amama atam	Value				
Model	Hyperparameter					
DeBERTa	Tensor Shape	[64, 36, 12, 64]				
	Tensor Rank	5				
	α	8				
	Learning Rate	1e-5				
	Tensor Shape	[64, 16, 9, 64]				
RoBERTa	Tensor Rank	4				
	α	1, 2, 8, 10				
	Learning Rate	1e-1				
LLaMA2-7B	Tensor Shape	[16, 16, 16, 16, 16, 16]				
	Tensor Rank	5				
	α	4				
	Learning Rate	1e-5				
LLaMA3-8B	Tensor Shape	[16, 4, 4, 4, 2, 2, 2, 2, 4, 4, 4, 16]				
	Tensor Rank	10				
	α	12				
	Learning Rate	1e-1				

LLaMA2-7B model fine-tuned with TT-LoRA. The heatmap in Figure 5 depicts the α value, signifying the magnitude of updates to the pre-trained weights during fine-tuning. As shown in Figure 5, model accuracy tends to decrease as the number of trainable parameters is reduced. In addition to model parameters, the value of α also influences the model accuracy. Nonetheless, TT-LoRA maintains superior or comparable performance to other PEFT methods, even with fewer parameters, demonstrating its capability to effectively minimize trainable parameters within strict hardware limits.

V. CONCLUSION

We introduced TT-LoRA, a parameter-efficient fine-tuning approach that leverages tensor train decomposition to significantly reduce the number of trainable parameters. TT-LoRA demonstrated substantial model compression and improved performance when fine-tuning BERT and LLaMA-based models across various tasks. Compared to other state-of-the-art PEFT methods, TT-LoRA achieved higher or comparable average accuracy with a significantly smaller model size, underscoring its effectiveness in both parameter reduction and performance enhancement. For future work, we aim to extend TT-LoRA to compress larger-scale models such as LLaMA3.1-405B, Grok 2.0, and Mistral Large. Additionally, we plan to explore the compression of additional layers within LLMs to achieve even greater levels of compression.

REFERENCES

- [1] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," arXiv preprint arXiv:2302.13971, 2023.
- [2] OpenAI et al., "Gpt-4 technical report," arXiv preprint arXiv:2303.08774, 2023.
- [3] S. Wu et al., "A comparative study of open-source large language models, gpt-4 and claude 2: Multiple-choice test taking in nephrology," arXiv preprint arXiv:2308.04709, 2023.
- [4] B. Goertzel and C. Pennachin, Artificial general intelligence. Springer, 2007, vol. 2.
- [5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, vol. 30, 2017.
- [6] M. U. Hadi, K. Y. Azeez, U. F. Mohammed, M. H. Farag, H. A. Omar, and M. Ahmed, "Large language models: a comprehensive survey of its applications, challenges, limitations, and future prospects," *Authorea Preprints*, 2023.

- [7] A. Faiz et al., "Limcarbon: Modeling the end-to-end carbon footprint of large language models," arXiv preprint arXiv:2309.14393, 2023.
- [8] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2790–2799.
- [9] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," arXiv preprint arXiv:2101.00190, 2021.
- [10] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," arXiv preprint arXiv:2104.08691, 2021
- [11] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, and L. Wang, "Lora: Low-rank adaptation of large language models," arXiv preprint arXiv:2106.09685, 2021.
- [12] Y. Yang, J. Zhou, N. Wong, and Z. Zhang, "Loretta: Low-rank economic tensor-train adaptation for ultra-low-parameter fine-tuning of large language models," arXiv preprint arXiv:2402.11417, 2024.
- [13] I. V. Oseledets, "Tensor-train decomposition," SIAM Journal on Scientific Computing, vol. 33, no. 5, pp. 2295–2317, 2011.
- [14] X. Ma et al., "A tensorized transformer for language modeling," in Advances in neural information processing systems, vol. 32, 2019.
- [15] A. Novikov et al., "Tensorizing neural networks," in Advances in neural information processing systems, vol. 28, 2015.
- [16] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *International conference on machine learning*. PMLR, 2019, pp. 2790–2799.
- [17] Z. Lin, A. Madotto, and P. Fung, "Exploring versatile generative language model via parameter-efficient transfer learning," arXiv preprint arXiv:2004.03829, 2020.
- [18] J. Pfeiffer, A. Kamath, A. Rücklé, K. Cho, and I. Gurevych, "Adapter-fusion: Non-destructive task composition for transfer learning," arXiv preprint arXiv:2005.00247, 2020.
- [19] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," arXiv preprint arXiv:2106.09685, 2021.
- [20] X. Liu, K. Ji, Y. Fu, W. L. Tam, Z. Du, Z. Yang, and J. Tang, "P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks," arXiv preprint arXiv:2110.07602, 2021.
- [21] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov, "Tensorizing neural networks," Advances in neural information processing systems, vol. 28, 2015.
- [22] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," arXiv preprint arXiv:1910.01108, 2019.
- [23] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced bert with disentangled attention," arXiv preprint arXiv:2006.03654, 2020.
- [24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," Advances in neural information processing systems, vol. 30, 2017.
- [25] A. Aghajanyan, L. Zettlemoyer, and S. Gupta, "Intrinsic dimensionality explains the effectiveness of language model fine-tuning," arXiv preprint arXiv:2012.13255, 2020.
- [26] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," SIAM review, vol. 51, no. 3, pp. 455–500, 2009.
- [27] M. Bhattarai et al., "Distributed non-negative tensor train decomposition," in 2020 IEEE HPEC. IEEE, 2020, pp. 1–7.
- [28] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," arXiv preprint arXiv:1907.11692, 2019.
- [29] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," arXiv preprint arXiv:1804.07461, 2018.
- [30] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "Superglue: A stickier benchmark for generalpurpose language understanding systems," *Advances in neural informa*tion processing systems, vol. 32, 2019.
- [31] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale et al., "Llama 2: Open foundation and fine-tuned chat models," arXiv preprint arXiv:2307.09288, 2023.
- [32] N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, S. Hu, Y. Chen, C.-M. Chan, W. Chen et al., "Parameter-efficient fine-tuning of large-

- scale pre-trained language models," Nature Machine Intelligence, vol. 5, no. 3, pp. 220-235, 2023.
- [33] E. B. Zaken, S. Ravfogel, and Y. Goldberg, "Bitfit: Simple parameterefficient fine-tuning for transformer-based masked language-models," arXiv preprint arXiv:2106.10199, 2021.
- [34] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.
- [35] L. Li, K. G. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: Bandit-based configuration evaluation for hyperparameter optimization." in ICLR (Poster), 2017, p. 53.
- [36] Meta, "Introducing llama: A foundational, 65-billion-parameter large language model," 2023, https://ai.meta.com/blog/ language large-language-model-llama-meta-ai/.
 [37] Ray, "How to use tune with pytorch," 2023, https://docs.ray.io/en/latest/
- tune/examples/tune-pytorch-cifar.html.