TeRA: Vector-based Random Tensor Network for High-Rank Adaptation of Large Language Models

Yuxuan Gu[*1], **Wuyang Zhou**[*†1], **Giorgos Iacovides**[1], **Danilo Mandic**[1]

[1]Department of Electrical and Electronic Engineering, Imperial College London
{yuxuan.gu21, wuyang.zhou19, giorgos.iacovides20, d.mandic}@imperial.ac.uk

Abstract

Parameter-Efficient Fine-Tuning (PEFT) methods, such as Low-Rank Adaptation (LoRA), have significantly reduced the number of trainable parameters needed in fine-tuning large language models (LLMs). The developments of LoRA-style adapters have considered two main directions: (1) enhancing model expressivity with high-rank adapters, and (2) aiming for further parameter reduction, as exemplified by vector-based methods. However, these approaches come with a trade-off, as achieving the expressivity of high-rank weight updates typically comes at the cost of sacrificing the extreme parameter efficiency offered by vector-based techniques. To address this issue, we propose a vector-based random **Te**nsor network for high-**R**ank **A**daptation (TeRA), a novel PEFT method that achieves high-rank weight updates while retaining the parameter efficiency of vector-based PEFT adapters. This is achieved by parametrizing the tensorized weight update matrix as a Tucker-like tensor network (TN), whereby large randomly initialized factors are frozen and shared across layers, while only small layer-specific scaling vectors, corresponding to diagonal entries of factor matrices, are trained. Comprehensive experiments demonstrate that TeRA matches or even outperforms existing high-rank adapters, while requiring as few trainable parameters as vector-based methods. Theoretical analysis and ablation studies validate the effectiveness of the proposed TeRA method. The code is available at https://github.com/guyuxuan9/TeRA.

1 Introduction

Foundation models, such as the Llama (Touvron et al., 2023; Team, 2024) and GPT (Brown et al., 2020; OpenAI, 2023) series, have revolutionized the field of natural language processing (NLP) by demonstrating strong generalization abilities across a diverse range of tasks. Although these models have been pre-trained on a large-scale corpus of textual data, supervised fine-tuning (SFT) is often necessary to improve their performance on specific downstream tasks. However, the large number of trainable parameters in full-parameter SFT can become computationally prohibitive in resource-constrained environments.

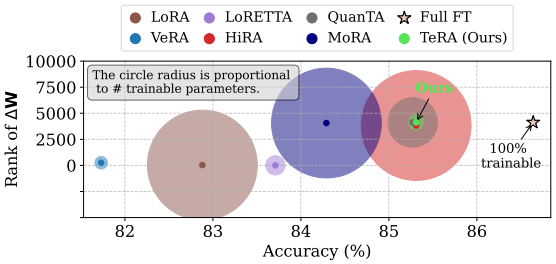

Figure 1: TeRA exhibits superior performance, high-rank and parameter efficiency trade-off. On commonsense reasoning task with Llama-3-8B, TeRA achieves high-rank updates and great performance, while keeping as few trainable parameters as vector-based methods (VeRA). Full Fine-tuning (FFT) requires many parameters, so its parameter count is omitted (See Table 2).

Parameter-Efficient Fine-Tuning (PEFT) methods, particularly Low-Rank Adaptation (LoRA) (Hu et al., 2022), have substantially reduced the number of trainable parameters in adapting large language models (LLMs) to downstream tasks. These methods operate by freezing the original pre-trained weights and training only the weight updates, $\Delta\mathbf{W} \in \mathbb{R}^{J_1 \times J_2}$, which are used to modify the original weights. The core assumption of LoRA is that these weight updates can be effectively approximated by a low-rank decomposition, $\Delta\mathbf{W} = \mathbf{AB}$, where $\mathbf{A} \in \mathbb{R}^{J_1 \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times J_2}$, where r controlls the rank upper bound.

However, the low rank assumption of $\Delta\mathbf{W}$ can limit its expressivity when adapting to more complex downstream tasks (Jiang et al., 2024;

[*]Equal Contribution.
[†]Project Lead.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2314–2329

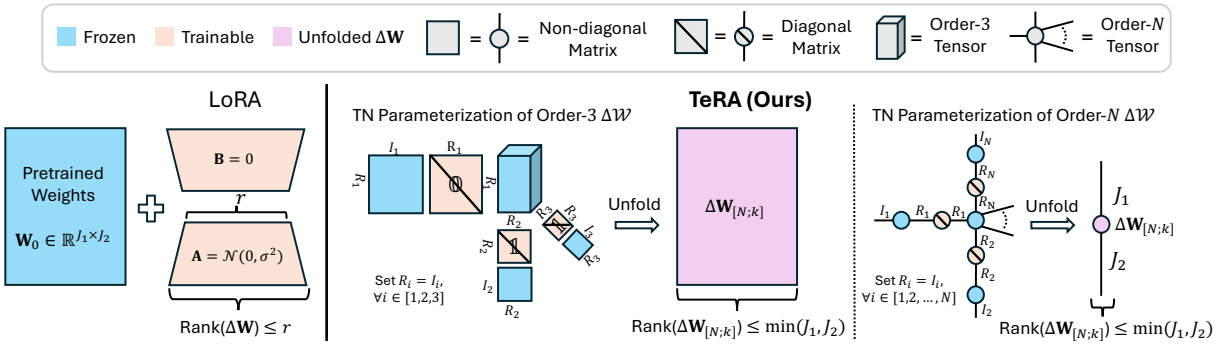

Figure 2: A comparison between the matrix-based LoRA (Hu et al., 2022) and our proposed tensor-based TeRA methods. LoRA represents the weight update matrix using two smaller matrices, while TeRA employs a Tucker-like (Tucker, 1964) multi-linear tensor network (TN) to parametrize the tensorized $\Delta\mathcal{W}$. This design allows TeRA to achieve high-rank updates with much fewer trainable parameters than LoRA.

Huang et al., 2025). To address this, Huang et al. (2025) proposed Hadamard High-Rank Adaptation (HiRA), which enhances update expressivity by introducing a Hadamard product between the learned weight update matrix, $\mathbf{AB}$, and the frozen pre-trained weight matrix, $\mathbf{W}_0$. Specifically, HiRA defines the high-rank weight update as $\Delta\mathbf{W} = (\mathbf{AB}) \odot \mathbf{W}_0$, where $\odot$ denotes the Hadamard product, and $\mathbf{W}_0$ is the frozen pre-trained weight matrix. Similar to LoRA, HiRA still requires training all parameters in $\mathbf{A}$ and $\mathbf{B}$.

Methods like VeRA (Kopiczko et al., 2024) reduce the trainable parameter count further by freezing $\mathbf{A}$ and $\mathbf{B}$, while training only two scaling vectors, $\mathbf{b}$ and $\mathbf{d}$, but they remain constrained by the low-rank assumption. Specifically, the VeRA weight update is parametrized as $\Delta\mathbf{W} = \mathbf{\Lambda}_b\mathbf{B}\mathbf{\Lambda}_d\mathbf{A}$, where $\mathbf{b} \in \mathbb{R}^{J_1}$ and $\mathbf{d} \in \mathbb{R}^{r}$ are the two trainable diagonal entries in $\mathbf{\Lambda}_b \in \mathbb{R}^{J_1 \times J_1}$ and $\mathbf{\Lambda}_d \in \mathbb{R}^{r \times r}$, respectively. Thus, VeRA requires only a fraction of the trainable parameters of LoRA.

It is possible to obtain high-rank weight updates with low-rank adapters. However, this leads to an explosion in the number of trainable parameters. Meanwhile, the low-rank assumption in the weight update matrices has been shown to restrict their performance in more complex tasks, such as arithmetic and reasoning (Huang et al., 2025). It is therefore natural to ask:

- *Is it possible to achieve high- (or full-) rank weight updates and their desired performance, while maintaining a similar number of trainable parameters to vector-based PEFT adapters such as VeRA?*

To resolve this trade-off, we propose a vector-based random **Te**nsor network for high-**R**ank **A**daptation (TeRA), a novel PEFT method that achieves high-rank weight updates using very few trainable parameters. The core idea is to tensorize the weight update matrix, $\Delta\mathbf{W}$, into a higher-order tensor, and then parametrize it using a Tucker-like (Tucker, 1964) tensor network (Cichocki et al., 2015), as shown in Figure 2. Within this tensor network, we freeze large randomly initialized factors, share them across all layers, and train only the small layer-specific scaling vectors that parametrize diagonal factor matrices. This design effectively decouples the rank of the update matrix from the number of trainable parameters, enabling high-rank adaptation with extremely low parameter counts similar to vector-based methods.

Extensive experiments demonstrate that TeRA establishes a superior trade-off between model performance, high rank, and parameter efficiency. As shown in Figure 1, TeRA matches the accuracy of HiRA using orders of magnitude fewer parameters. Compared to parameter count-matched methods like VeRA and Low-Rank Economic Tensor-Train Adaptation (LoRETTA) (Yang et al., 2024), TeRA offers a significant accuracy improvement. This stems from the more expressive high-rank weight updates in TeRA across all model layers (See Figure 3), a property that low-rank methods inherently lack. TeRA maintains near full-rank updates across all layers, enabling more expressive adaptations.

In summary, our contributions are as follows:

- We propose TeRA, a new PEFT method that uses a multi-linear Tucker-like tensor network to parametrize the tensorized high-rank weight updates, which can be merged with the original weights at inference time and incur zero computational and latency overhead.

- A theoretical analysis is provided demonstrating that TeRA can achieve high-rank weight updates with fewer parameters than existing methods. Our analysis formalizes the trade-off between the performance and the trainable parameter count of TeRA.

- Extensive experiments compare TeRA with baseline methods, demonstrating that TeRA exhibits superior performance while requiring a similar number of trainable parameters to vector-based PEFT adapters.

2 Related work

Prompt-based Methods. One category of PEFT methods comprises prompt-based methods, such as Prompt Tuning (Lester et al., 2021) and P-Tuning (Liu et al., 2022), which introduce additional trainable virtual tokens into the input of LLMs and optimize only these tokens. These methods are sensitive to initialization schemes and require additional computational costs during inference.

Low-rank Adaptation (LoRA). Introduced by Hu et al. (2022), LoRA employs two matrices, $\mathbf{A} \in \mathbb{R}^{J_1 \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times J_2}$, to parametrize the weight update matrix, $\Delta\mathbf{W} = \mathbf{AB} \in \mathbb{R}^{J_1 \times J_2}$, as a low-rank decomposition, thereby significantly reducing the number of trainable parameters. Since $\Delta\mathbf{W}$ has the same dimensionality as the pre-trained weight matrix, it can be merged into the pre-trained weight during inference, eliminating any additional inference overhead. Based on LoRA, VeRA (Kopiczko et al., 2024) proposes to randomly initialize and freeze the $\mathbf{A}$ and $\mathbf{B}$ matrices, share them across layers, and train only two scaling vectors, $\mathbf{b}$ and $\mathbf{d}$, thus significantly reducing the trainable parameters needed for low-rank weight updates. Other variants of LoRA impose different algebraic structures within $\Delta\mathbf{W}$ to achieve parameter reduction or high-rank updates (Zhang et al., 2023; Dettmers et al., 2023; Liu et al., 2024; Borse et al., 2024).

Recently, tensor-based methods, which operate on tensorized neural network weights (Gu et al., 2025; Zhou et al., 2026), have also been shown to be effective in fine-tuning LLMs (Yang et al., 2024; Bershatsky et al., 2024). More specifically, they focus on reducing the number of trainable parameters compared to LoRA by assuming higher-order low-rank update structures (Oseledets, 2011), which may still fail to capture high-rank updates for complex tasks (Huang et al., 2025).

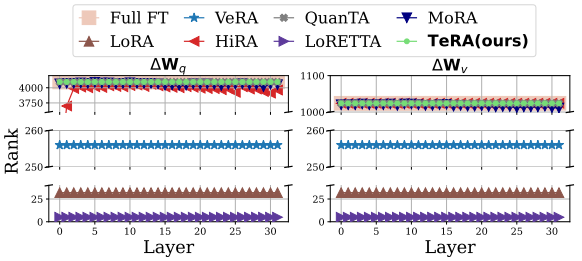

Figure 3: Rank analysis of $\Delta\mathbf{W}_q$ (max allowed rank of 4096) and $\Delta\mathbf{W}_v$ (max allowed rank of 1024) across Llama-3-8B layers. TeRA consistently maintains a high (near-full) rank. In contrast, methods like LoRA and VeRA have lower-rank weight updates, limiting their expressivity. See Figure 7 in Appendix K for zoomed-in comparison of high-rank adapters.

High-rank Adaptation. To overcome the limited expressivity of low-rank adaptation, high-rank variants of LoRA have been proposed. QuanTA (Chen et al., 2024) uses a quantum-inspired method to efficiently achieve high-rank weight updates. MoRA (Jiang et al., 2024) employs a square matrix to achieve high-rank updates, while HiRA (Huang et al., 2025) uses the Hadamard product to learn a high-rank weight update matrix. Different from these methods, TeRA effectively achieves high-rank weight updates and uses much fewer trainable parameters, similar amount to that in vector-based PEFT adapters such as VeRA.

3 Tensors and Multi-linear Algebra Preliminaries

The mathematical notations used in this paper are listed in Table 1. This is consistent with the notation used in Cichocki et al. (2015).

Symbol	Meaning
a, $\mathbf{a}$, $\mathbf{A}$, $\mathcal{A}$	Scalar, vector, matrix, tensor
$(\cdot)^\top$	Matrix transpose
$(\cdot)^\dagger$	Matrix pseudoinverse
$\|\cdot\|_F$	Frobenius norm
$\mathcal{A}(i_1, \ldots, i_N)$	The $(i_1, \ldots, i_N)$-th element of $\mathcal{A}$
$\mathcal{A} \times_n \mathbf{B}$	Mode-n product
$\text{diag}(\mathbf{a})$	A diagonal matrix whose diagonal is $\mathbf{a}$

Table 1: Mathematical notations

A tensor is a multi-dimensional array and a higher-order generalization of vectors and matrices, whereby a vector, $\mathbf{a} \in \mathbb{R}^{I_1}$ is an order-1 tensor, while a matrix, $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$, is an order-2 tensor. An order-N tensor is denoted by $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$.

Tensorization and Matricization. Tensorization (folding) is the process to fold a lower-dimensional tensor into a higher-dimensional one. A matrix $\mathbf{A} \in \mathbb{R}^{J_1 \times J_2}$ can be folded into an order-N tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, provided that $\prod_{i=1}^{k} I_i = J_1$ and $\prod_{i=k+1}^{N} I_i = J_2$ for some $k \in [1, N]$. Its inverse operation is termed matricization (unfolding). Unfolding operation converts an order-N tensor, $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, into a matrix, $\mathbf{A}_{[N;k]} \in \mathbb{R}^{\prod_{i=1}^{k} I_i \times \prod_{i=k+1}^{N} I_i}$, whose elements are given by the following

$$\mathbf{A}_{[N;k]}(\overline{i_1 \cdots i_k}, \overline{i_{k+1} \cdots i_N}) = \mathcal{A}(i_1, i_2, \ldots, i_N), \tag{1}$$

The corresponding tensorization operation is denoted by $\text{Fold}_{[N;k]}(\mathbf{A}_{[N;k]}) = \mathcal{A}$.

Mode-n product. Mode-n product between a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a matrix $\mathbf{B} \in \mathbb{R}^{I_n \times J_n}$ yields a tensor $\mathcal{C} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N}$. This operation is denoted as

$$\mathcal{C} = \mathcal{A} \times_n \mathbf{B}, \tag{2}$$

and its element-wise definition form is

$$\mathcal{C}(i_1, \cdots, i_{n-1}, j_n, i_{n+1}, \cdots, i_N) = \sum_{i_n=1}^{I_N} \mathcal{A}(i_1, \cdots, i_{n-1}, i_n, i_{n+1}, \cdots, i_N)\mathbf{B}(i_n, j_n). \tag{3}$$

Tucker Decomposition. Tucker decomposition (Tucker, 1964) is a generalization of Singular Value Decomposition (SVD) to higher-order tensors (De Lathauwer et al., 2000a) and a cornerstone of multi-linear tensor network (Cichocki et al., 2015; Zhou et al., 2024). Given an order-N tensor, $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, Tucker decomposition expresses it using a smaller order-N core tensor $\mathcal{G} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$, where $R_i \ll I_i$, and N factor matrices $\{\mathbf{B}^{(i)} \in \mathbb{R}^{R_i \times I_i}\}_{i=1}^{N}$, in the form

$$\mathcal{X} = \mathcal{G} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \times_3 \cdots \times_N \mathbf{B}^{(N)}. \tag{4}$$

Identifying the optimal set of Tucker ranks, $[R_1, \ldots, R_N]$, efficiently is an active area of research (Zhou et al., 2025), with numerous recent studies focusing on advanced methods for tensor rank search (Iacovides et al., 2024; Li et al., 2023; Iacovides et al., 2025).

4 Methodology

TeRA parametrizes the tensorized weight update matrix $\Delta\mathbf{W}$ using a Tucker-like tensor network, as shown in Figure 2. The method involves two steps: (1) Tensorize the weight update matrix into a higher-order tensor $\Delta\mathcal{W} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$; (2) Parametrize this tensor using a Tucker-like tensor network with majority of large factor matrices frozen, and only small diagonal matrices trainable. After training, $\Delta\mathcal{W}$ is unfolded to obtain $\Delta\mathbf{W}_{[N;k]}$.

Specifically, TeRA tensorizes a weight update matrix into an order-N tensor $\Delta\mathcal{W} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} = \text{Fold}_{[N;k]}(\Delta\mathbf{W}_{[N;k]} \in \mathbb{R}^{J_1 \times J_2})$, where $\prod_{i=1}^{k} I_i = J_1$, $\prod_{i=k+1}^{N} I_i = J_2$, $I_i \geq 2 \; \forall \; i = 1, \ldots, N$, and $1 \leq k < N$. For example, the attention weight matrices in Llama-2-7B have size 4096×4096, which can be tensorized into shapes of $64 \times 64 \times 64 \times 64$, $16 \times 16 \times \cdots \times 16$, $4 \times 4 \times \cdots \times 4$, etc.

TeRA Formulation. TeRA parametrizes the weight update tensor, $\Delta\mathcal{W}$, as the mode-n product of a frozen core tensor $\mathcal{G} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$, a set of N frozen non-diagonal factor matrices $\{\mathbf{A}^{(i)} \in \mathbb{R}^{R_i \times I_i}\}_{i=1}^{N}$, and a set of N trainable vectors $\{\mathbf{d}^{(i)} \in \mathbb{R}^{R_i}\}_{i=1}^{N}$, which are diagonal entries of $\{\text{diag}(\mathbf{d}^{(i)}) \in \mathbb{R}^{R_i \times R_i}\}_{i=1}^{N}$. The resulting parametrization is given by

$$\begin{aligned}\Delta\mathcal{W} = \mathcal{G} \times_1 \text{diag}(\mathbf{d}^{(1)}) \times_2 \text{diag}(\mathbf{d}^{(2)}) \times_3 \\ \cdots \times_N \text{diag}(\mathbf{d}^{(N)}) \times_1 \mathbf{A}^{(1)} \\ \times_2 \mathbf{A}^{(2)} \times_3 \cdots \times_N \mathbf{A}^{(N)}.\end{aligned} \tag{5}$$

During fine-tuning, the core tensor $\mathcal{G}$ and the factor matrices $\{\mathbf{A}^{(i)}\}_{i=1}^{N}$ are randomly initialized and kept frozen. These matrices are shared between all the adapted layers of the LLM. The only trainable components are the diagonal entries of the matrices $\{\text{diag}(\mathbf{d}^{(i)})\}_{i=1}^{N}$. All $\text{diag}(\mathbf{d}^{(i)})$ are initialized as identity matrices, except for one $\text{diag}(\mathbf{d}^{(i)})$, which is initialized as a zero matrix to ensure $\Delta\mathbf{W}_{[N;k]}$ is zero at initialization. This reduces the number of trainable parameters to just $\sum_{i=1}^{N} R_i$ per TeRA adapter, where the ranks $[R_1, \ldots, R_N]$ are hyperparameters.

Notably, TeRA introduces zero computational overhead during inference. After fine-tuning, the TeRA weight update $\Delta\mathbf{W}_{[N;k]}$ is unfolded from $\Delta\mathcal{W}$ and added to the pre-trained weights, $\mathbf{W}_0$, to yield the final weights

$$\mathbf{W}_{\text{final}} = \mathbf{W}_0 + \Delta\mathbf{W}_{[N;k]}. \tag{6}$$

Theorem 1. *Let $\Delta\mathbf{W} \in \mathbb{R}^{J_1 \times J_2}$ be the weight update matrix, and $\Delta\mathcal{W} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} = \text{Fold}_{[N;k]}(\Delta\mathbf{W}_{[N;k]})$ be its folded weight update tensor, parametrized by TeRA as in Eq. (5). The following inequality holds*

$$\text{rank}\left(\Delta\mathbf{W}_{[N;k]}\right) \leq \min\left(\prod_{i=1}^{k} R_i, \prod_{i=k+1}^{N} R_i\right). \tag{7}$$

This allows for a full-rank update matrix under any tensorization (folding) schemes if $R_i = I_i \; \forall i = 1, \ldots, N$, i.e., $\text{rank}\left(\Delta\mathbf{W}_{[N;k]}\right) \leq \min\left(J_1, J_2\right)$.

This shows that TeRA not only can enable high-rank adaptation, but also requires a very small number of trainable parameters. For example, to allow for a full-rank weight update matrix of size $J_1 \times J_2$ ($J_1 \geq J_2$), we need $J_1 \cdot J_2 + J_2 \cdot J_2$ trainable parameters in LoRA and at *least* $J_1 + J_2$ trainable parameters in both VeRA and HiRA. However, TeRA only requires $\sum_{i=1}^{N} I_i$ trainable parameters, whereby $\prod_{i=1}^{k} I_i = J_1$, $\prod_{i=k+1}^{N} I_i = J_2$, $I_i \geq 2 \; \forall i = 1, \ldots, N$, and $1 \leq k < N$.

Theorem 2. *For a full-rank weight update matrix, TeRA is more parameter-efficient than VeRA and HiRA, i.e., when $R_i = I_i, \forall i = 1, \ldots, N$, the following holds*

$$\sum_{i=1}^{N} R_i \leq J_1 + J_2, \tag{8}$$

where $\prod_{i=1}^{k} R_i = J_1$, $\prod_{i=k+1}^{N} R_i = J_2$, $R_i \geq 2 \; \forall i = 1, \ldots, N$, and $1 \leq k < N$.

Theorem 2 establishes that TeRA is provably more parameter-efficient than existing methods such VeRA and HiRA when a full-rank weight update matrix is considered. Specifically, TeRA needs at *most* $J_1 + J_2$ trainable parameters to parametrize a full-rank update matrix. The number of trainable parameters used in TeRA can also be further reduced by tensorizing the weight matrix to higher-dimensions. For example, a 4096×4096 matrix can be tensorized to a $64 \times 64 \times 64 \times 64$ tensor. A full-rank update with HiRA or VeRA would require at least $4096 + 4096 = 8192$ parameters, while TeRA requires only $64 + 64 + 64 + 64 = 256$ parameters. By increasing the tensor order N to 24 (e.g., tensor size of $2 \times 2 \times \cdots \times 2$), the number of trainable parameters in TeRA can be reduced to as few as $2 \times 24 = 48$.

Theorem 3. *Consider the optimal weight update $\mathbf{W}^\star \in \mathbb{R}^{J_1 \times J_2}$ and the TeRA weight update, $\mathbf{W}_{TeRA} \in \mathbb{R}^{J_1 \times J_2}$, whose tensorized format is defined in Eq. (5). Denote $\bigotimes_{i=1}^{k} \mathbf{A}^{(i)}$ by $\mathbf{L}^\top \in \mathbb{R}^{\prod_{i=1}^{k} R_i \times J_1}$, $\bigotimes_{i=k+1}^{N} \mathbf{A}^{(i)}$ by $\mathbf{M} \in \mathbb{R}^{J_2 \times \prod_{i=k+1}^{N} R_i}$, and $\mathbf{Z} = \mathbf{L}^\dagger \mathbf{W}^\star \mathbf{M}^\dagger \oslash \mathbf{G}_{[N;k]}$, where $\oslash$ is the element-wise division. Then, the following inequality holds*

$$\begin{aligned}
&\min_{\{\text{diag}(\mathbf{d}^{(i)})\}_{i=1}^{N}} \|\mathbf{W}^\star - \mathbf{W}_{TeRA}\|_F^2 \\
&\leq \|\mathbf{W}^\star - \mathbf{L}\mathbf{L}^\dagger \mathbf{W}^\star \mathbf{M}^\dagger \mathbf{M}\|_F^2 \\
&+ g_{max}^2 (\|\mathbf{Z}\|_F^2 - \|\,\text{Fold}_{[N;k]}(\mathbf{Z})\|_2^2) \|\mathbf{L}\|_F^2 \|\mathbf{M}\|_F^2,
\end{aligned} \tag{9}$$

where g_{max} is the largest entry in $\mathcal{G}$, and $\|\cdot\|_2$ denotes the tensor spectral norm.

Theorem 3 provides a theoretical upper bound on the approximation error between the optimal weight update and the TeRA weight update. This bound consists of two terms. The first term, $\|\mathbf{W}^\star - \mathbf{L}\mathbf{L}^\dagger \mathbf{W}^\star \mathbf{M}^\dagger \mathbf{M}\|_F^2$, quantifies the portion of the optimal update $\mathbf{W}^\star$, which lies outside the subspace characterized by the frozen factor matrices $\{\mathbf{A}^{(i)}\}_{i=1}^{N}$. This error can only be minimized by expanding the subspace through increasing the ranks, $\{R_i\}_{i=1}^{N}$. Consequently, the bound provides direct theoretical motivation to maximize the ranks to their corresponding tensor dimension sizes, *i.e.*, $R_i = I_i, \; \forall i = 1, 2, \ldots, N$. As a side benefit, this also reduces the number of hyperparameters in TeRA by eliminating the need to choose the tensor network ranks $[R_1, R_2, \ldots, R_N]$.

The second error term establishes the trade-off between parameter efficiency and approximation accuracy (expressivity) in TeRA. It is bounded by a term dependent on the tensor spectral norm of $\mathbf{Z}$ (De Lathauwer et al., 2000b). The tensor spectral norm has been shown to decrease as the order of the tensor, N, increases (Wang et al., 2017). Therefore, assuming $R_i = I_i \; \forall i = 1, \ldots, N$, this indicates that using a higher-order tensorization (a larger N in $\Delta\mathcal{W} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$) reduces the number of trainable parameters ($\sum_i^N R_i$), but enlarges the upper bound of the approximation error. Conversely, lower-order tensorization may result in a tighter error bound, but at the cost of more trainable parameters. The proofs for the three theorems are provided in the appendix.

Model	Method	Params (%)	BoolQ	PIQA	SIQA	ARC-c	ARC-e	OBQA	HellaS	WinoG	Average
Llama-2-7B	Full FT	100	73.8	84.2	81.0	94.7	85.2	88.9	75.6	84.8	83.53
	Prompt Tuning	0.0012	55.93	12.35	30.50	6.06	8.63	9.40	6.91	40.57	21.29
	P-Tuning	0.7428	58.75	36.02	0.20	0.17	1.98	0.80	0.01	0.00	12.24
	LoRA	0.2484	67.65	79.22	78.20	69.20	83.88	78.60	81.05	80.98	77.35
	HiRA (r=1)	0.0078	65.35	77.97	72.42	62.03	81.48	64.80	79.90	70.01	71.74
	HiRA (r=32)	0.2484	**69.39**	**83.24**	78.86	**71.33**	**86.57**	**81.40**	**87.23**	81.69	**79.97**
	MoRA (r=32)	0.2484	65.17	81.12	77.74	67.75	84.47	75.8	84.12	79.64	76.97
	QuanTA	0.0408	66.88	81.28	78.45	70.82	85.40	79.4	75.86	82.16	77.53
	LoRETTA	0.0052	67.52	79.33	76.31	65.70	84.34	76.00	86.32	80.98	77.06
	VeRA	0.0041	64.59	78.84	76.56	65.44	83.63	73.20	82.71	77.90	75.36
	TeRA (Ours)	**0.0039**	65.78	81.23	**78.92**	69.45	84.05	81.00	85.94	**82.64**	78.63
Llama-3-8B	Full FT	100	75.4	88.0	81.8	96.5	89.3	93.1	83.0	86.0	86.64
	Prompt Tuning	0.0010	56.85	45.05	36.13	31.57	32.74	29.20	14.01	50.12	36.96
	P-Tuning	0.6240	59.97	11.64	8.19	7.42	8.63	9.60	1.77	37.65	18.11
	LoRA	0.1695	71.99	85.91	79.58	76.19	88.55	82.60	92.54	85.63	82.88
	HiRA (r=1)	0.0053	68.04	85.75	75.54	76.96	90.32	77.00	89.04	77.43	80.01
	HiRA (r=32)	0.1695	**73.09**	88.85	81.06	80.38	92.68	86.20	94.37	85.87	**85.31**
	MoRA (r=32)	0.1692	69.05	88.25	80.14	80.89	**92.72**	84.2	94.09	85.00	84.29
	QuanTA	0.0343	70.03	88.41	81.67	81.05	92.21	85.6	94.68	88.47	85.27
	LoRETTA	0.0045	65.17	**89.01**	79.53	79.86	91.54	84.40	94.40	84.69	83.58
	VeRA	**0.0022**	67.95	85.64	76.51	77.22	91.29	81.00	91.96	82.24	81.73
	TeRA (Ours)	0.0033	70.70	88.08	**81.58**	**80.89**	92.00	**88.00**	**94.92**	**86.27**	**85.31**

Table 2: Accuracy comparison of different PEFT methods on the Commonsense170k dataset. Full FT performance is cited from (Liu et al., 2025). The best and second best values among LoRA-style PEFT adapters are highlighted in bold and underlined, respectively.

5 Experiments

We conducted extensive experiments to demonstrate the effectiveness of TeRA across a diverse set of reasoning and generation tasks in English. We also performed a series of ablation studies to validate our model design and analyze the impact of hyperparameter choices.

Implementation Details. We used two LLMs as the base models for fine-tuning: Llama-2-7B and Llama-3-8B. The percentage of trainable parameters was calculated as $\frac{\text{\# trainable params}}{\text{\# total params}}$, where *trainable parameters* refers to those requiring gradient updates, and *total parameters* include both the frozen and trainable parameters across all layers in the LLM. In practice, we find that tensorizing only one dimension of the weight update matrix in TeRA yields a good trade-off between performance and the number of trainable parameters. To reduce the hyperparameters search space, we always tensorize each dimension into equal-sized modes and set $R_i = I_i \; \forall i = 1, \ldots, N$. E.g., a dimension of size 4096 can be tensorized into 64×64, $16 \times 16 \times 16 \times 16$, etc. We report the average performance of TeRA over 5 independent runs.

Baseline Methods. We benchmark TeRA against two main categories of PEFTs: prompt-based methods (Prompt-Tuning, P-Tuning) and LoRA-style adapters with no inference overhead (LoRA, HiRA, MoRA, QuanTA, LoRETTA, VeRA). To ensure fairness in terms of number of trainable parameters, following (Kopiczko et al., 2024), we applied all methods to the query and value weights in the attention modules. We also report HiRA with two rank settings, $r \in \{1, 32\}$, to show how it performs with different number of trainable parameters.

5.1 Commonsense Reasoning

Following Huang et al. (2025), we evaluated TeRA on eight commonsense reasoning tasks using the Commonsense170k benchmark (Hu et al., 2023), which has $170,420$ query-answer pairs. The eight sub-tasks include: BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2020), ARC-e and ARC-c (Clark et al., 2018), and OBQA (Mihaylov et al., 2018). The test accuracy is given in Table 2.

Results. Table 2 reveals that TeRA consistently outperforms baseline methods which require similar number of trainable parameters, such as HiRA ($r = 1$), LoRETTA, and VeRA, in terms of average accuracy. Specifically, TeRA achieved an average accuracy of 78.63% on Llama-2-7B (followed by 77.06% with LoRETTA) and 85.31% on Llama-3-8B (followed by 83.58% with LoRETTA). TeRA matched the performance of the best performing high-rank adapter, HiRA ($r = 32$), while having

Model	Method	Params (%)	BLEU	BERT F1	BERT-R	BERT-P	Meteor	R-L	Average
Llama-2-7B	Prompt Tuning	0.0012	0.04	72.44	77.38	68.23	0.80	0.80	36.62
	P-Tuning	0.7428	0.60	83.29	83.33	83.28	15.11	12.36	46.33
	LoRA	0.2484	2.59	85.25	85.31	**85.21**	14.14	13.28	47.63
	HiRA ($r = 1$)	0.0078	2.39	84.84	84.93	84.78	13.35	12.60	47.15
	HiRA ($r = 32$)	0.2484	**2.67**	85.26	85.37	85.17	13.94	13.19	47.60
	MoRA ($r = 32$)	0.2484	1.91	84.85	84.91	84.83	11.80	11.59	46.65
	QuanTA	0.0408	2.65	85.14	85.32	85.00	14.19	13.32	47.60
	LoRETTA	0.0052	2.22	84.99	84.97	85.03	12.37	11.85	46.91
	VeRA	0.0041	2.27	85.01	85.04	85.02	12.64	12.55	47.09
	TeRA (Ours)	**0.0039**	2.62	**85.28**	**85.40**	85.19	**14.26**	**13.39**	**47.69**
Llama-3-8B	Prompt Tuning	0.0010	1.45	82.99	82.99	83.05	14.72	13.13	46.39
	P-Tuning	0.6240	1.50	81.52	81.07	82.01	15.49	13.55	45.86
	LoRA	0.1695	3.24	85.02	84.49	**85.60**	15.16	14.14	47.94
	HiRA ($r = 1$)	0.0053	3.36	84.81	84.40	85.26	15.12	14.19	47.86
	HiRA ($r = 32$)	0.1695	3.22	84.58	84.35	84.87	14.90	13.72	47.61
	MoRA ($r = 32$)	0.1692	2.35	84.20	83.79	84.67	11.96	11.51	46.41
	QuanTA	0.0343	3.04	84.55	84.19	84.97	13.90	13.08	47.29
	LoRETTA	0.0045	2.97	84.77	84.26	85.31	13.78	13.15	47.37
	VeRA	0.0022	3.12	84.61	84.27	85.01	14.56	13.72	47.55
	TeRA (Ours)	**0.0021**	**3.38**	**85.03**	**84.53**	85.57	**15.32**	**14.59**	**48.07**

Table 3: Evaluation results of different PEFT methods on the ConvAI2 dataset. The considered metrics include BLEU, BERTScore (F1/R/P), Meteor, and ROUGE-L.

$64\times$ fewer trainable parameters in the Llama-2-7B model and $51\times$ less trainable parameters in the Llama-3-8B model. Remarkably, TeRA requires fewer parameters than even the lowest-rank HiRA configuration ($r = 1$), yet consistently delivering superior performance. These results demonstrate that TeRA achieves the performance benefits of high-rank adapters, while offering significantly improved parameter efficiency.

High-Rank Weight Updates of TeRA. To visualize the high-rank nature of TeRA, Figure 3 shows the ranks of the update matrices across different layers obtained in the commonsense reasoning task for Llama-3-8B. Observe that TeRA consistently obtained high-rank updates. Additionally, the weight updates of TeRA often reach full-rank, validating Theorem 1 empirically. In comparison, the low-rank weight updates of methods such as LoRA, LoRETTA, and VeRA may limit their expressivity.

5.2 Personalized Dialogue Generation

We evaluated the ability of fine-tuned models to engage in natural conversations using the ConvAI2 dataset (Dinan et al., 2019) with 17, 878 training and 1, 000 testing multi-turn conversations. Following the experimental setup of (Huang et al., 2025), where only the speaker's persona is revealed (self-persona setting), we report the quality of the generated responses using standard metrics, including BLEU, BERTScore (Zhang et al., 2020), METEOR (Banerjee and Lavie, 2005), and ROUGE (Lin, 2004) in Table 3.

Results. Table 3 shows that TeRA consistently achieved the highest average score among all baseline methods in conversational tasks and used the least number of trainable parameters among LoRA-style adapters. More specifically, TeRA achieved an average score of 47.69% with Llama-2-7B and 48.07% with Llama-3-8B. This shows the superior performance and parameter efficiency of TeRA in conversation fine-tuning tasks.

5.3 Arithmetic Reasoning

We evaluated arithmetic reasoning capabilities of the fine-tuned model through the Math10k dataset from (Cobbe et al., 2021; Koncel-Kedziorski et al., 2016; Ling et al., 2017), consisting of 10, 000 mathematical reasoning examples. We considered two sub-tasks: AQuA (Ling et al., 2017) and SVAMP (Patel et al., 2021). The test accuracies are reported in Table 4.

Results. TeRA consistently outperformed all baseline methods in AQuA and SVAMP while requiring the least number of trainable parameters. As shown in Table 4, with the Llama–2-7B backbone, TeRA achieved accuracies of 24.41% on AQuA and 49.7% on SVAMP, thus outperforming the strongest baseline by 5.51% and 1.7% absolute points, respectively. With Llama–3-8B, TeRA obtains 30.71% on AQuA and 73.1% on SVAMP,

Model	Method	Params (%)	AQuA	SVAMP
Llama -2-7B	HiRA ($r = 1$)	0.0078	18.90	27.40
	HiRA ($r = 32$)	0.2484	18.90	46.50
	LoRETTA	0.0052	12.60	47.50
	VeRA	0.0041	18.50	26.60
	TeRA (Ours)	**0.0039**	**24.41**	**49.70**
Llama -3-8B	HiRA ($r = 1$)	0.0053	27.95	57.70
	HiRA ($r = 32$)	0.1695	29.92	72.80
	LoRETTA	0.0045	18.50	56.50
	VeRA	0.0022	25.59	55.70
	TeRA (Ours)	**0.0021**	**30.71**	**73.10**

Table 4: Performance comparison in terms of test accuracy on arithmetic reasoning datasets.

outperforming HiRA ($r = 32$) by 0.79% and 0.3% absolute points while using $80\times$ fewer parameters.

5.4 Ablation study

Impact of Tensorization on Performances. Different tensorizations of the weight matrices lead to different trade-offs between the number of trainable parameters and the model performance, as formalized in Theorem 3. Figure 4 (left) shows that when both dimensions of the original weight matrix are tensorized, the performance decreases rapidly with decrease in the number of trainable parameters. Empirically, we achieve an optimal trade-off between performance and parameter efficiency by tensorizing only one dimension of the original weight matrix. Under this approach, the performance of TeRA remains robust across different tensorization sizes (See Figure 4 (right)).

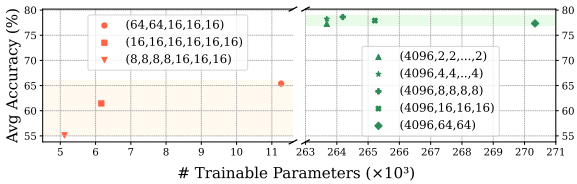

Figure 4: Average accuracies across eight commonsense reasoning tasks against number of trainable parameters under different tensorization strategies in Llama-2-7B.

Impact of Tensorization on Ranks. The high-rank property of TeRA is insensitive to the specific choice of tensorization schemes, as formalized in Theorem 1. We evaluated the ranks of the query weight updates, $\Delta\mathbf{W}_q$, and the value weight updates, $\Delta\mathbf{W}_v$, across different layers under various tensorizations. Figure 5 shows that, as desired, TeRA obtains high-rank (near full-rank) updates across different tensorization choices.

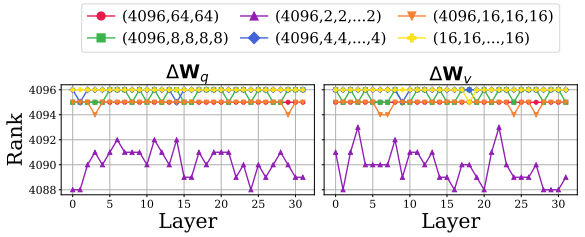

Figure 5: Ranks of $\Delta\mathbf{W}_q$ and $\Delta\mathbf{W}_v$ (Largest possible rank = 4096) in Llama-2-7B under different tensorization schemes in the commonsense reasoning task.

Initialization of Frozen Factor Matrices. To explore different initialization choices for the frozen factor matrices, we compared TeRA with its variant, TeRA$_{iden}$, where its frozen factor matrices are all identity matrices. Note that TeRA$_{iden}$ has the same number of trainable parameters as TeRA. Figure 6 shows that, with identical hyperparameters, TeRA consistently outperforms TeRA$_{iden}$ in terms of average accuracy, highlighting the effectiveness of the random tensor network initialization scheme in TeRA. We also experimented with non-sharing factor tensors and report the result in Appendix J.

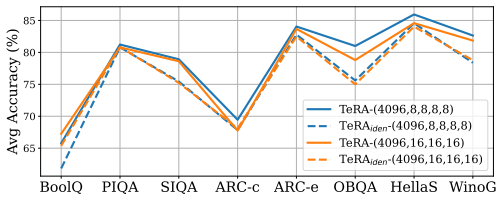

Figure 6: Comparison between TeRA and TeRA$_{iden}$ on the commonsense reasoning dataset with Llama-2-7B.

6 Conclusion

We have introduced TeRA, a high-rank PEFT adapter which employs a tensor network approach to parametrize the tensorized weight updates. Such a parametrization has allowed TeRA to offer a more effective alternative to existing vector-based adapters with a similar amount of trainable parameters, while achieving much better performances and high-rank updates. TeRA is particularly well-suited for large-scale customization scenarios, where hundreds of thousands of task- or user-specific adapters may be required. Its ability to maintain high expressivity while using extremely few trainable parameters per adapter makes it an attractive solution for scalable personalization of LLMs.

7 Limitations

Despite the good performance of TeRA and its low number of trainable parameters, the training process can be time-consuming due to the tensor contractions required for fine-tuning Large Language Models (LLMs). A promising direction for future work is the integration of TeRA with custom hardware specifically designed for accelerating these tensor operations. Furthermore, TeRA currently does not employ low-bit quantization methods. Incorporating such techniques could further reduce the memory footprint, making the method more accessible in resource-constrained settings (Zhou et al., 2023).

Moreover, although Theorem 3 has provided the theoretical foundation for significantly minimizing the search space of selecting the best tensorization scheme, identifying the optimal configuration for a given task still involves sampling different settings. Future work could focus on developing efficient methods for the selection of tensorization schemes for compressing neural network weights. We believe that our work can pave the way for future explorations in these avenues.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Daniel Bershatsky, Daria Cherniuk, Talgat Daulbaev, Aleksandr Mikhalev, and Ivan Oseledets. 2024. LoTR: Low Tensor Rank Weight Adaptation. *Preprint*, arXiv:2402.01376.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: Reasoning about Physical Commonsense in Natural Language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press.

Shubhankar Borse, Shreya Kadambi, Nilesh Prasad Pandey, Kartikeya Bhardwaj, Viswanath Ganapathy, Sweta Priyadarshi, Risheek Garrepalli, Rafael Esteves, Munawar Hayat, and Fatih Porikli. 2024. FouRA: Fourier Low-Rank Adaptation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language Models are Few-shot Learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Zhuo Chen, Rumen Dangovski, Charlotte Loh, Owen M Dugan, Di Luo, and Marin Soljacic. 2024. QuanTA: Efficient High-Rank Fine-Tuning of LLMs with Quantum-Informed Tensor Adaptation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Andrzej Cichocki, Danilo Mandic, Lieven De Lathauwer, Guoxu Zhou, Qibin Zhao, Cesar Caiafa, and Huy Anh Phan. 2015. Tensor Decompositions for Signal Processing Applications From Two-way to Multiway Component Analysis. *IEEE Signal Processing Magazine*, 32(2):145–163.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*.

Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. 2000a. A Multilinear Singular Value Decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278.

Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. 2000b. On the Best Rank-1 and Rank-(R1,R2,. . .,RN) Approximation of Higher-Order Tensors. *SIAM journal on Matrix Analysis and Applications*, 21(4):1324–1342.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander H. Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan

Lowe, Shrimai Prabhumoye, Alan W. Black, Alexander I. Rudnicky, Jason D. Williams, Joelle Pineau, Mikhail Burtsev, and Jason Weston. 2019. The Second Conversational Intelligence Challenge (ConvAI2). In *The NeurIPS'18 Competition: From Machine Learning to Intelligent Conversations*, pages 187–208. Springer.

Yuxuan Gu, Wuyang Zhou, Giorgos Iacovides, and Danilo Mandic. 2025. TensorLLM: Tensorising Multi-Head Attention for Enhanced Reasoning and Compression in LLMs. *IEEE International Joint Conference on Neural Networks (IJCNN 2025)*.

Kaiming He, 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*, pages 1026–1034.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. 2023. LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5254–5276.

Qiushi Huang, Tom Ko, Zhan Zhuang, Lilian Tang, and Yu Zhang. 2025. HiRA: Parameter-Efficient Hadamard High-Rank Adaptation for Large Language Models. In *The Thirteenth International Conference on Learning Representations*.

Giorgos Iacovides, Wuyang Zhou, Chao Li, Qibin Zhao, and Danilo Mandic. 2025. Domain-Aware Tensor Network Structure Search. *arXiv preprint arXiv:2505.23537*.

Giorgos Iacovides, Wuyang Zhou, and Danilo Mandic. 2024. Towards LLM-guided Efficient and Interpretable Multi-linear Tensor Network Rank Selection. *arXiv preprint arXiv:2410.10728*.

Ting Jiang, Shaohan Huang, Shengyue Luo, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, Qi Zhang, Deqing Wang, and Fuzhen Zhuang. 2024. MoRA: High-Rank Updating for Parameter-Efficient Fine-Tuning. *Preprint*, arXiv:2405.12130.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A Math Word Problem Repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California. Association for Computational Linguistics.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M Asano. 2024. VeRA: Vector-based Random Matrix Adaptation. In *The Twelfth International Conference on Learning Representations*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059. Association for Computational Linguistics.

Chao Li, Junhua Zeng, Chunmei Li, Cesar Caiafa, and Qibin Zhao. 2023. Alternating Local Enumeration (TnALE): Solving Tensor Network Structure Search with Fewer Evaluations. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Shiwei Li, Xiandi Luo, Haozhao Wang, Xing Tang, Ziqiang Cui, Dugang Liu, Yuhua Li, xiuqiang He, and Ruixuan Li. 2025. Beyond Higher Rank: Token-wise Input-Output Projections for Efficient Low-Rank Adaptation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program Induction by Rationale Generation: Learning to Solve and Explain Algebraic Word Problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-Tuning: Prompt Tuning Can Be Comparable to Fine-tuning Across Scales and Tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland. Association for Computational Linguistics.

Zihang Liu, Tianyu Pang, Oleg Balabanov, Chaoqun Yang, Tianjin Huang, Lu Yin, Yaoqing Yang, and Shiwei Liu. 2025. LIFT the Veil for the Truth: Principal Weights Emerge after Rank Reduction for Reasoning-Focused Supervised Fine-Tuning. In *Forty-second International Conference on Machine Learning*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. PiSSA: Principal Singular Values and Singular Vectors Adaptation of Large Language Models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

OpenAI. 2023. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*.

Ivan V Oseledets. 2011. Tensor-Train Decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP Models really able to Solve Simple Math Word Problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094. Association for Computational Linguistics.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. SocialIQA: Commonsense Reasoning about Social Interactions. In *Conference on Empirical Methods in Natural Language Processing*.

Llama Team. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*.

Ryota Tomioka and Taiji Suzuki. 2014. Spectral Norm of Random Tensors. *arXiv preprint arXiv:1407.1870*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and Fine-tuned Chat Models. *arXiv preprint arXiv:2307.09288*.

Ledyard R Tucker. 1964. The Extension of Factor Analysis to Three-Dimensional Matrices. *Contributions to mathematical psychology*, 110119:110–182.

Miaoyan Wang, Khanh Dao Duc, Jonathan Fischer, and Yun S Song. 2017. Operator Norm Inequalities between Tensor Unfoldings on the Partition Lattice. *Linear algebra and its applications*, 520:44–66.

Frank Wilcoxon. 1945. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83.

Yifan Yang, Jiajun Zhou, Ngai Wong, and Zheng Zhang. 2024. LoRETTA: Low-Rank Economic Tensor-Train Adaptation for Ultra-Low-Parameter Fine-Tuning of Large Language Models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3161–3176.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a Machine Really Finish Your Sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. In *The Eleventh International Conference on Learning Representations*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*.

Jiajun Zhou, Jiajun Wu, Yizhao Gao, Yuhao Ding, Chaofan Tao, Boyu Li, Fengbin Tu, Kwang-Ting Cheng, Hayden Kwok-Hay So, and Ngai Wong. 2023. DyBit: Dynamic Bit-Precision Numbers for Efficient Quantized Neural Network Inference. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 43(5):1613–1617.

Wuyang Zhou, Yuxuan Gu, Giorgos Iacovides, and Danilo Mandic. 2026. KromHC: Manifold-Constrained Hyper-Connections with Kronecker-Product Residual Matrices. *arXiv preprint arXiv:2601.21579*.

Wuyang Zhou, Giorgos Iacovides, Kriton Konstantinidis, Ilya Kisil, and Danilo Mandic. 2025. Understanding the Rank of Tensor Networks via an Intuitive Example-Driven Approach. *arXiv preprint arXiv:2507.10170*.

Wuyang Zhou, Yu-Bang Zheng, Qibin Zhao, and Danilo Mandic. 2024. Tensor Star Tensor Decomposition and Its Applications to Higher-order Compression and Completion. *arXiv preprint arXiv:2403.10481*.

A Preliminaries

The mathematical notations used in Appendix are listed in Table 5. This is consistent with the notation used in Cichocki et al. (2015). See also the Preliminaries section of the main paper.

Symbol	Meaning
a, $\mathbf{a}$, $\mathbf{A}$, $\mathcal{A}$	Scalar, vector, matrix, tensor
$(\cdot)^\top$	Matrix transpose
$(\cdot)^\dagger$	Matrix pseudoinverse
$\|\cdot\|_F$	Frobenius norm
$\mathcal{A}(i_1, i_2, \ldots, i_N)$	The $(i_1, \ldots, i_N)$-th element of $\mathcal{A}$
$\mathbf{a} \circ \mathbf{b}$	Outer product between two vectors
$\mathbf{A} \otimes \mathbf{B}$	Kronecker product
$\mathcal{A} \times_n \mathbf{B}$	Mode-n product
$\text{diag}(\mathbf{a})$	Diagonal matrix with diagonal as $\mathbf{a}$
$\text{Fold}_{[N;k]}(\mathbf{A}_{[N;k]})$	Fold matrix $\mathbf{A}_{[N;k]}$ to tensor $\mathcal{A}$

Table 5: Mathematical notations

B Theorem 1 and Its Proof

TeRA Formulation. We parametrize the weight update tensor $\Delta\mathcal{W}$ as the mode-n product of a frozen core tensor $\mathcal{G} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$, a set of N frozen non-diagonal factor matrices $\{\mathbf{A}^{(i)} \in \mathbb{R}^{R_i \times I_i}\}_{i=1}^N$, and a set of N trainable vectors $\{\mathbf{d}^{(i)} \in \mathbb{R}^{R_i}\}_{i=1}^N$, which are diagonal entries of $\{\text{diag}(\mathbf{d}^{(i)}) \in \mathbb{R}^{R_i \times R_i}\}_{i=1}^N$. The definition of TeRA is

$$\begin{aligned}\Delta\mathcal{W} = \mathcal{G} \times_1 \text{diag}(\mathbf{d}^{(1)}) \times_2 \text{diag}(\mathbf{d}^{(2)}) \times_3 \\ \cdots \times_N \text{diag}(\mathbf{d}^{(N)}) \times_1 \mathbf{A}^{(1)} \\ \times_2 \mathbf{A}^{(2)} \times_3 \cdots \times_N \mathbf{A}^{(N)}.\end{aligned} \tag{10}$$

or equivalently

$$\begin{aligned}&\Delta\mathcal{W}(i_1, \ldots, i_N) \\ &= \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_N=1}^{R_N} \mathcal{G}(r_1, r_2, \ldots, r_N) \\ &\quad \mathbf{d}^{(1)}(r_1)\mathbf{d}^{(2)}(r_2) \cdots \mathbf{d}^{(N)}(r_N) \\ &\quad \mathbf{A}^{(1)}(r_1, i_1)\mathbf{A}^{(2)}(r_2, i_2) \cdots \mathbf{A}^{(N)}(r_N, i_N)\end{aligned} \tag{11}$$

Theorem 1. *Let $\Delta\mathbf{W} \in \mathbb{R}^{J_1 \times J_2}$ be the weight update matrix, and $\Delta\mathcal{W} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} = \text{Fold}_{[N;k]}(\Delta\mathbf{W}_{[N;k]})$ be its folded weight update tensor, parametrized by TeRA as in Eq. (10). The following inequality holds*

$$\text{rank}\left(\Delta\mathbf{W}_{[N;k]}\right) \leq \min\left(\prod_{i=1}^{k} R_i, \prod_{i=k+1}^{N} R_i\right). \tag{12}$$

This allows for a full-rank update matrix under any tensorization (folding) schemes if $R_i = I_i \; \forall i = 1, \ldots, N$, i.e., $\text{rank}\left(\Delta\mathbf{W}_{[N;k]}\right) \leq \min\left(J_1, J_2\right)$.

Proof. We first group the diagonal and non-diagonal factor matrices that correspond to the same mode and rewrite Eq. (10) as a standard Tucker decomposition (Tucker, 1964)

$$\Delta\mathcal{W} = \mathcal{G} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)}, \tag{13}$$

where $\mathbf{B}^{(i)} := \text{diag}(\mathbf{d}^{(i)})\mathbf{A}^{(i)} \in \mathbb{R}^{R_i \times I_i}$. Since $\mathbf{A}^{(i)}$ for $i = 1, \ldots, N$ are randomly initialized using Kaiming initialization (He et al., 2015), they are usually full-rank, making $\mathbf{B}^{(i)}$ also full-rank given no zero entries in $\mathbf{d}^{(i)}$.

Recall that the tensorization was chosen such that

$$J_1 = \prod_{i=1}^{k} I_i, \qquad J_2 = \prod_{i=k+1}^{N} I_i. \tag{14}$$

Using the Kronecker product, Eq. (13) can be written in a matrix format as

$$\Delta\mathbf{W}_{[N;k]} = \left(\bigotimes_{i=1}^{k} \mathbf{B}^{(i)}\right)^\top \mathbf{G}_{[N;k]} \left(\bigotimes_{i=k+1}^{N} \mathbf{B}^{(i)}\right), \tag{15}$$

where $\otimes$ denotes the Kronecker product, and $\mathbf{G}_{(1:N,k)} \in \mathbb{R}^{\left(\prod_{i=1}^{k} R_i\right) \times \left(\prod_{i=k+1}^{N} R_i\right)}$ is a matricization of the core tensor.

Therefore, the following holds

$$\begin{aligned}\text{rank}\left(\Delta\mathbf{W}_{[N;k]}\right) \leq \min\Bigg\{ &\text{rank}\left(\bigotimes_{\mathbf{i=1}}^{\mathbf{k}} \mathbf{B}^{(\mathbf{i})}\right)^\top, \\ &\text{rank}\left(\mathbf{G}_{[\mathbf{N};\mathbf{k}]}\right), \text{rank}\left(\bigotimes_{\mathbf{i=k+1}}^{\mathbf{N}} \mathbf{B}^{(\mathbf{i})}\right)\Bigg\}.\end{aligned} \tag{16}$$

As ranks, R_i, are set to be smaller or equal to than their corresponding mode sizes, I_i, $\prod_{i=1}^{k} I_i \geq \prod_{i=1}^{k} R_i$ and $\prod_{i=k+1}^{N} I_i \geq \prod_{i=k+1}^{N} R_i$. Therefore,

$$\text{rank}(\Delta\mathbf{W}_{[N;k]}) \leq \min\Big(\prod_{i=1}^{k} R_i, \prod_{i=k+1}^{N} R_i\Big). \tag{17}$$

Thus, if $I_i = R_i$ for $i = 1, \ldots, N$

$$\begin{aligned}\text{rank}\left(\Delta\mathbf{W}_{[N;k]}\right) &\leq \min\Big(\prod_{i=1}^{k} I_i, \prod_{i=k+1}^{N} I_i\Big) \\ &= \min\left(J_1, J_2\right).\end{aligned} \tag{18}$$

□

C Theorem 2 and Its Proof

Theorem 2. *For a full-rank weight update matrix, TeRA is more parameter-efficient than VeRA and HiRA, i.e., when $R_i = I_i, \forall i = 1, \ldots, N$, the following holds*

$$\sum_{i=1}^{N} R_i \leq J_1 + J_2, \tag{19}$$

where $\prod_{i=1}^{k} R_i = J_1$, $\prod_{i=k+1}^{N} R_i = J_2$, $R_i \geq 2\ \forall i = 1, \ldots, N$, and $1 \leq k < N$.

Proof. Since $R_i = I_i, \forall i = 1, \ldots, N$, we just need to prove that $I_1 + I_2 + \cdots + I_k \leq J_1$ for $I_1 \times I_2 \times \cdots \times I_k = J_1$. Then, we naturally have $I_1 + I_2 + \cdots + I_N \leq J_1 + J_2$, such that $I_1 \times I_2 \times \cdots \times I_k = J_1$, $I_{k+1} \times I_{k+2} \times \cdots \times I_N = J_2$, $\{I_i \geq 2\}_{i=1}^{N}$, and $1 \leq k < N$.

When $N = k = 1$, $I_1 = J_1$, thus $I_1 = J_1$. The induction hypothesis is such that assume for some $N = k \geq 1$, we have $\sum_{i=1}^{k} I_i \leq \prod_{i=1}^{k} I_i$. For $a, b \geq 2$, $(a-1)(b-1) \geq 1$. Therefore, $a \cdot b \geq a + b$. This in turn yields for $N = k+1 \geq 1$, $I_{k+1} + \sum_{i=1}^{k} I_i \leq I_{k+1} \cdot \sum_{i=1}^{k} I_i$, as $\sum_{i=1}^{k} I_i, I_{k+1} \geq 2$. Therefore, we have proven that $\sum_{i=1}^{k+1} I_i \leq \prod_{i=1}^{k+1} I_i$. As this holds for $k+1$, by mathematical induction, the following holds for $k \geq 1$, $I_1 + I_2 + \cdots + I_k \leq J_1$ where $I_1 \times I_2 \times \cdots \times I_k = J_1$. Applying this result twice, we have $I_1 + I_2 + \cdots + I_N \leq J_1 + J_2$. Therefore, $\sum_{i=1}^{N} R_i \leq J_1 + J_2$.

□

D Theorem 3 and Its Proof

Theorem 3. *Consider the optimal weight update $\mathbf{W}^\star \in \mathbb{R}^{J_1 \times J_2}$ and the TeRA weight update, $\mathbf{W}_{TeRA} \in \mathbb{R}^{J_1 \times J_2}$, whose tensorized format is defined in Eq. (5). Denote $\bigotimes_{i=1}^{k} \mathbf{A}^{(i)}$ by $\mathbf{L}^\top \in \mathbb{R}^{\prod_{i=1}^{k} R_i \times J_1}$, $\bigotimes_{i=k+1}^{N} \mathbf{A}^{(i)}$ by $\mathbf{M} \in \mathbb{R}^{J_2 \times \prod_{i=k+1}^{N} R_i}$, and $\mathbf{Z} = \mathbf{L}^\dagger \mathbf{W}^\star \mathbf{M}^\dagger \oslash \mathbf{G}_{[N;k]}$, where $\oslash$ is the element-wise division. Then, the following inequality holds*

$$\begin{aligned}
&\min_{\{\mathrm{diag}(\mathbf{d}^{(i)})\}_{i=1}^{N}} \|\mathbf{W}^\star - \mathbf{W}_{TeRA}\|_F^2 \\
&\leq \|\mathbf{W}^\star - \mathbf{L}\mathbf{L}^\dagger \mathbf{W}^\star \mathbf{M}^\dagger \mathbf{M}\|_F^2 \\
&+ g_{max}^2 (\|\mathbf{Z}\|_F^2 - \|\,\mathrm{Fold}_{[N;k]}(\mathbf{Z})\|_2^2) \|\mathbf{L}\|_F^2 \|\mathbf{M}\|_F^2,
\end{aligned} \tag{20}$$

where g_{max} is the largest entry in $\mathcal{G}$, and $\|\cdot\|_2$ denotes the tensor spectral norm.

Proof. In the analysis, we assume that there is no zero in $\mathbf{G}_{[N;k]}$. Let $\mathbf{W}^\star \in \mathbb{R}^{J_1 \times J_2}$ be the optimal weight update and $\mathbf{W}_{TeRA}$ be the learned TeRA update. Denote $\bigotimes_{i=1}^{k} \mathbf{A}^{(i)}$ by $\mathbf{L}^\top \in \mathbb{R}^{\prod_{i=1}^{k} R_i \times J_1}$, and $\bigotimes_{i=k+1}^{N} \mathbf{A}^{(i)}$ by $\mathbf{M} \in \mathbb{R}^{J_2 \times \prod_{i=k+1}^{N} R_i}$.

Using Kronecker product, Eq. (13) can be written in matrix format as

$$\begin{aligned}
\mathbf{W}_{TeRA} = &\left(\bigotimes_{i=1}^{k} \mathrm{diag}(\mathbf{d}^{(i)}) \mathbf{A}^{(i)} \right)^\top \mathbf{G}_{[N;k]} \\
&\left(\bigotimes_{i=k+1}^{N} \mathrm{diag}(\mathbf{d}^{(i)}) \mathbf{A}^{(i)} \right),
\end{aligned} \tag{21}$$

Let $\mathbf{E} = \bigotimes_{i=1}^{k} \mathrm{diag}(\mathbf{d}^{(i)})$ and $\mathbf{F} = \bigotimes_{i=k+1}^{N} \mathrm{diag}(\mathbf{d}^{(i)})$. Using the Kronecker identity, we can write

$$\mathbf{W}_{TeRA} = \mathbf{L}\mathbf{E}\mathbf{G}_{[N;k]}\mathbf{F}\mathbf{M}, \tag{22}$$

Thus, our aim is to minimize the following discrepancy measured in the Frobenius norm

$$\|\mathbf{W}^\star - \mathbf{L}\mathbf{E}\mathbf{G}_{[N;k]}\mathbf{F}\mathbf{M}\|_F^2. \tag{23}$$

As $\mathbf{W}^\star - \mathbf{L}\mathbf{E}\mathbf{G}_{[N;k]}\mathbf{F}\mathbf{M} = \mathbf{W}^\star - \mathbf{L}\mathbf{L}^\dagger\mathbf{W}^\star\mathbf{M}^\dagger\mathbf{M} + \mathbf{L}\mathbf{L}^\dagger\mathbf{W}^\star\mathbf{M}^\dagger\mathbf{M} - \mathbf{L}\mathbf{E}\mathbf{G}_{[N;k]}\mathbf{F}\mathbf{M}$, we have

$$\begin{aligned}
&\|\mathbf{W}^\star - \mathbf{L}\mathbf{E}\mathbf{G}_{[N;k]}\mathbf{F}\mathbf{M}\|_F^2 \\
&\leq \|\mathbf{W}^\star - \mathbf{L}\mathbf{L}^\dagger\mathbf{W}^\star\mathbf{M}^\dagger\mathbf{M}\|_F^2 \\
&\quad + \|\mathbf{L}\mathbf{L}^\dagger\mathbf{W}^\star\mathbf{M}^\dagger\mathbf{M} - \mathbf{L}\mathbf{E}\mathbf{G}_{[N;k]}\mathbf{F}\mathbf{M}\|_F^2
\end{aligned} \tag{24}$$

Given fixed ranks, $\{R_i\}_{i=1}^{N}$, the first term in Eq. (24) is the irreducible error from training, which lies outside the subspace characterized by the frozen non-diagonal factor matrices. The second term in Eq. (24) is the in-subspace error which can be minimized by training $\mathbf{E}$ and $\mathbf{F}$. We can also write the following inequality for the second term in Eq. (24)

$$\begin{aligned}
&\min_{\mathbf{E},\mathbf{F}} \|\mathbf{L}\mathbf{L}^\dagger\mathbf{W}^\star\mathbf{M}^\dagger\mathbf{M} - \mathbf{L}\mathbf{E}\mathbf{G}_{[N;k]}\mathbf{F}\mathbf{M}\|_F^2 \\
&= \min_{\mathbf{E},\mathbf{F}} \|\mathbf{L}(\mathbf{L}^\dagger\mathbf{W}^\star\mathbf{M}^\dagger - \mathbf{E}\mathbf{G}_{[N;k]}\mathbf{F})\mathbf{M}\|_F^2 \\
&\leq \min_{\mathbf{E},\mathbf{F}} \|\mathbf{L}\|_F^2 \|\mathbf{M}\|_F^2 \|\mathbf{L}^\dagger\mathbf{W}^\star\mathbf{M}^\dagger - \mathbf{E}\mathbf{G}_{[N;k]}\mathbf{F})\|_F^2
\end{aligned} \tag{25}$$

Additionally, since $\mathbf{E}$ and $\mathbf{F}$ are both diagonal matrices, $\mathbf{E}\mathbf{G}_{[N;k]}\mathbf{F} = \mathbf{G}_{[N;k]} \odot (\mathrm{diag}(\mathbf{E}) \circ \mathrm{diag}(\mathbf{F}))$, where $\odot$ and $\circ$ denotes the

Target Modules	Params (%)	BoolQ	PIQA	SIQA	ARC-c	ARC-e	OBQA	HellaS	WinoG	Average
Query only	0.0020	65.54	78.13	72.06	60.07	78.62	67.4	79.62	75.06	72.06
Value only	0.0020	62.94	80.36	77.69	67.75	82.49	78.2	84.52	81.61	76.94
Query & Value	0.0040	**65.78**	**81.23**	**78.92**	**69.45**	**84.05**	**81.00**	**85.94**	**82.64**	**78.63**

Table 6: The difference between applying TeRA only on query matrices, value matrices, and both query and value matrices in Llama-2-7B on the commonsense reasoning dataset. The tensorization scheme is $(4096, 8, 8, 8, 8)$.

Hadamard product and the outer-product, respectively. The term, $\mathbf{E}\mathbf{G}_{[N;k]}\mathbf{F}$, can be written as $\mathbf{G}_{[N;k]} \odot \operatorname{diag}(\mathbf{E}) \circ \operatorname{diag}(\mathbf{F})$, where $\odot$ denotes the Hadamard product, $\circ$ denotes the outer product, and $\operatorname{diag}(\cdot)$ is the vector containing the diagonal of the input matrix. Thus, we can rewrite Eq. (25) as

$$\begin{aligned}&\min_{\mathbf{E},\mathbf{F}} \|\mathbf{L}\|_F^2\|\mathbf{M}\|_F^2\|\mathbf{L}^\dagger\mathbf{W}^\star\mathbf{M}^\dagger - \mathbf{E}\mathbf{G}_{[N;k]}\mathbf{F})\|_F^2 \\ &\leq \min_{\mathbf{E},\mathbf{F}} g_{max}^2\|\mathbf{L}\|_F^2\|\mathbf{M}\|_F^2 \\ &\quad \|\mathbf{L}^\dagger\mathbf{W}^\star\mathbf{M}^\dagger \oslash \mathbf{G}_{[N;k]} - \operatorname{diag}(\mathbf{E}) \circ \operatorname{diag}(\mathbf{F})\|_F^2 \end{aligned} \tag{26}$$

where g_{max} is the largest entry in $\mathcal{G}$, and $\oslash$ denotes element-wise division. Denote $\mathbf{Z} = \mathbf{L}^\dagger\mathbf{W}^\star\mathbf{M}^\dagger \oslash \mathbf{G}_{[N;k]}$, to arrive at

$$\begin{aligned}&\min_{\mathbf{E},\mathbf{F}} \|\mathbf{Z} - \operatorname{diag}(\mathbf{E}) \circ \operatorname{diag}(\mathbf{F})\|_F^2 \\ &= \min_{\{\mathbf{d}^{(i)}\}_{i=1}^N} \|\operatorname{Fold}_{[N;k]}(\mathbf{Z}) - \mathbf{d}^{(1)} \circ \\ &\qquad \mathbf{d}^{(2)} \circ \cdots \circ \mathbf{d}^{(N)}\|_F^2 \end{aligned} \tag{27}$$

Let $\operatorname{Fold}_{[N;k]}(\hat{\mathbf{Z}})$ be the best rank-1 approximation of $\operatorname{Fold}_{[N;k]}(\mathbf{Z})$. Then, Eq. (27) is equivalent to the minimization of the approximation error between $\operatorname{Fold}_{[N;k]}(\hat{\mathbf{Z}})$ and $\operatorname{Fold}_{[N;k]}(\mathbf{Z})$, which is given by (De Lathauwer et al., 2000b) as

$$\begin{aligned}&\min_{\{\mathbf{d}^{(i)}\}_{i=1}^N} \|\operatorname{Fold}_{[N;k]}(\mathbf{Z}) - \operatorname{Fold}_{[N;k]}(\hat{\mathbf{Z}})\|_F^2 \\ &= \|\operatorname{Fold}_{[N;k]}(\mathbf{Z})\|_F^2 - \|\operatorname{Fold}_{[N;k]}(\mathbf{Z})\|_2^2 \end{aligned} \tag{28}$$

where $\|\cdot\|_2^2$ denotes the tensor spectral norm (Tomioka and Suzuki, 2014). More specfically, $\|\operatorname{Fold}_{[N;k]}(\mathbf{Z})\|_2^2$ is defined as,

$$\begin{aligned}&\|\operatorname{Fold}_{[N;k]}(\mathbf{Z})\|_2^2 \\ &= \sup_{\{\mathbf{u}^{(i)}\}_{i=1}^N} \operatorname{Fold}_{[N;k]}(\mathbf{Z}) \times_1 \mathbf{u}^{(1)} \times_2 \cdots \times_N \mathbf{u}^{(N)} \end{aligned} \tag{29}$$

where $\{\mathbf{u}^{(i)}\}_{i=1}^N$ are unit-norm vectors.

Combining Eq. (24), (25), (26), and (28), we arrive at

$$\begin{aligned}&\min_{\{\operatorname{diag}(\mathbf{d}^{(i)})\}_{i=1}^N} \|\mathbf{W}^\star - \mathbf{W}_{TeRA}\|_F^2 \\ &\leq \|\mathbf{W}^\star - \mathbf{L}\mathbf{L}^\dagger\mathbf{W}^\star\mathbf{M}^\dagger\mathbf{M}\|_F^2 \\ &\quad + g_{max}^2(\|\mathbf{Z}\|_F^2 - \|\operatorname{Fold}_{[N;k]}(\mathbf{Z})\|_2^2)\|\mathbf{L}\|_F^2\|\mathbf{M}\|_F^2. \end{aligned} \tag{30}$$

□

E Ablation Study on TeRA Placements in Transformers

We explored different placements of TeRA on the attention weight matrices. Specifically, we compared three cases: 1) Applying TeRA only on the query matrices; 2) only on the value matrices; 3) on both the query and the value matrices. As shown in Table 6, applying TeRA on both the query and the value matrices yields the best performance.

F Hyperparameters

All experiments were conducted using one NVIDIA A100 (80GB) GPU. The AdamW optimizer (Loshchilov and Hutter, 2019) was employed with 100 warm-up steps. The general hyperparameters for TeRA for all three datasets mentioned in the main paper are listed in Table 9. The frozen factors in TeRA are initialized with Kaiming init (He et al., 2015).

The dimension used for splitting the two models across the three datasets were chosen from the last two rows in Table 9, where `indim` represents the dimension of which the input feature is split into, while `outdim` denotes the dimension of which the output feature is split into. For example, in `indim=4096` and `outdim=8` setting, the weight update matrix of shape (4096,4096) is unfolded from a tensor of shape (4096,8,8,8,8).

For the other methods reported in the paper, we chose r according to the recommendations in the original paper. Specifically, LoRA uses a rank of 32; LoRETTA employs LoRETTA$_{\text{rep}}$ with a base rank of 32 and a TT-rank of 8; and VeRA uses a rank of 256.

Model	Method	Params (%)	BoolQ	PIQA	SIQA	ARC-c	ARC-e	OBQA	HellaS	WinoG	Average
Llama-2-7B	LoRA (r=32)	0.2484	67.65	79.22	78.20	69.20	83.88	78.60	81.05	80.98	77.35
	LoRA (r=128)	0.9861	69.76	83.62	79.38	72.61	86.53	81.80	88.57	84.77	80.88
	HiRA (r=1)	0.0078	65.35	77.97	72.42	62.03	81.48	64.80	79.90	70.01	71.74
	HiRA (r=32)	0.2484	69.39	83.24	78.86	71.33	86.57	81.40	87.23	81.69	79.97
	HiRA (r=128)	0.9861	68.59	82.43	80.04	71.42	85.94	79.80	87.30	81.37	79.61
	TeRA (Ours)	**0.0039**	65.78	81.23	78.92	69.45	84.05	81.00	85.94	82.64	78.63
Llama-3-8B	LoRA (r=32)	0.1695	71.99	85.91	79.58	76.19	88.55	82.60	92.54	85.63	82.88
	LoRA (r=128)	0.6744	73.61	87.87	82.55	81.74	92.21	87.8	95.59	88.24	86.20
	HiRA (r=1)	0.0053	68.04	85.75	75.54	76.96	90.32	77.00	89.04	77.43	80.01
	HiRA (r=32)	0.1695	73.09	88.85	81.06	80.38	92.68	86.20	94.37	85.87	85.31
	HiRA (r=128)	0.6744	74.31	90.10	81.83	82.51	92.97	88.4	95.92	88.16	86.78
	TeRA (Ours)	**0.0033**	70.70	88.08	81.58	80.89	92.00	88.00	94.92	86.27	85.31

Table 7: Performance and parameter efficiency of TeRA compared with higher-rank LoRA/HiRA.

Model	Method	BoolQ	PIQA	SIQA	ARC-c	ARC-e	OBQA	HellaS	WinoG	Average
Llama-2-7B	TeRA	**65.78**	**81.23**	**78.92**	**69.45**	**84.05**	**81.00**	**85.94**	**82.64**	**78.63**
	TeRA$_{unique}$	61.47	0.16	0	0.09	0.04	0	0.15	0	7.74
Llama-3-8B	TeRA	**70.70**	**88.08**	**81.58**	**80.89**	**92.00**	**88.00**	**94.92**	**86.27**	**85.31**
	TeRA$_{unique}$	58.47	66.27	29.99	16.30	14.35	3.60	11.86	11.92	26.59

Table 8: Performance comparison of TeRA and TeRA$_{unqiue}$, where the frozen Tucker core and factor matrices are unique for each finetuned matrix in each layer.

Hyperparameter	Value
Optimizer	AdamW
Weight Decay	0
Base Model	[Llama-2-7B, Llama-3-8B]
Learning Rate	[1e-3, 2e-4, 3e-4]
Warm Up	100 steps
Batch Size	32
Target Modules	q_proj, v_proj
Evaluation Steps	Every 80 steps
indim	[2,4,8,16,64,4096]
outdim	[2,4,8,16,64,4096]

Table 9: Hyperparameters for TeRA.

G Computational Resource Details

All experiments were conducted using one NVIDIA A100 (80GB) GPU on a Linux operation system. Linux operating system was used. Pytorch version used was 2.2.1. We used Tranformers 4.41.1, and PEFT 0.17.1 from Huggingface.

H Statistical Significances

Wilcoxon signed-rank test (Wilcoxon, 1945) was used to test whether TeRA statistically significantly outperforms methods with similar number of parameters, such as HiRA ($r = 1$) (Huang et al., 2025), VeRA (Kopiczko et al., 2024) and LoRETTA (Yang et al., 2024), across the three datasets tested. TeRA performed statistically better than HiRA ($r = 1$) with a p-value of 3.97×10^{-7}. TeRA performed statistically better than VeRA with a p-value of 3.97×10^{-7}. TeRA performed statistically better than LoRETTA with a p-value of 2.20×10^{-5}. TeRA also performed statistically better than the LoRA with a p-value of 2.37×10^{-4}. We also used Wilcoxon signed-rank test to test whether HiRA ($r = 32$) performed statistically better than TeRA. The results confirm that HiRA ($r = 32$) did not perform statistically better or worse than TeRA.

I LoRA and HiRA with higher ranks

To highlight the superior parameter efficiency and accuracy trade-off of TeRA, we evaluated LoRA and HiRA with higher ranks. The results in Table 7 show that TeRA achieved comparable performance, while requiring about 200$\times$ fewer trainable parameters.

J Ablation on Sharing Tucker Core and Factor Matrices

To illustrate the necessity of using a shared Tucker core and factor matrices across all layers, we compared TeRA with its variant TeRA$_{unique}$, where a unique frozen Tucker core $\mathcal{G}$ and factor matrices $\{\mathbf{A}^{(i)}\}_{i=1}^{N}$ were used for each fine-tuned matrix in each layer. As shown in Table 8, a significant performance degradation is observed for TeRA$_{unique}$ under the same experimental setup as TeRA.

Model	Method	Params (%)	BoolQ	PIQA	SIQA	ARC-c	ARC-e	OBQA	HellaS	WinoG	Average
Llama-3-8B	TopLoRA ($r = 16$)	0.1369	**72.60**	**88.30**	80.09	**81.14**	91.92	85.20	94.31	**86.42**	85.00
	PiSSA ($r = 16$)	0.0848	67.33	85.09	79.84	75.42	88.55	80.60	91.96	85.24	81.76
	DoRA ($r = 32$)	0.1715	58.65	85.36	80.45	76.11	89.27	82.8	92.33	85.56	81.32
	TeRA	**0.0033**	70.70	88.08	**81.58**	80.89	**92.00**	**88.00**	**94.92**	86.27	**85.31**

Table 10: Performance comparison of TopLoRA, PiSSA, DoRA, and TeRA.

This degradation might be due to the loss of a consistent subspace across layers. When the Tucker core and factor matrices are shared, each layer projects its weight update into the same subspace through its trainable diagonal matrices. As a result, the updates from different layers can reinforce or complement each other. In contrast, without sharing, the updates in each layer are confined to independent random subspaces. Since only the diagonal matrices are trainable, each layer must search for effective directions in isolation, which reduces efficiency and makes optimization more difficult.

K Rank Comparison between High-Rank Adapters

Figure 7 shows the detailed comparison of ranks among high-rank methods.

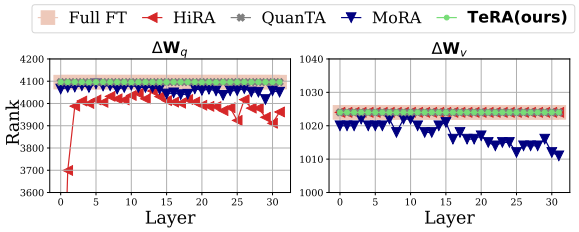

Figure 7: Rank analysis of $\Delta\mathbf{W}_q$ (max allowed rank of 4096) and $\Delta\mathbf{W}_v$ (max allowed rank of 1024) across Llama-3-8B layers for high-rank adapters.

L Additional experiments

The performance comparison of more PEFT methods, TopLoRA (Li et al., 2025), such as PiSSA (Meng et al., 2024) and DoRA (Liu et al., 2024), is illustrated in Table 10. TeRA achieves the best overall performance, while using the least number of trainable parameters, demonstrating again the superior performance and parameter-efficeincy trade-off.

M Artifact License and Others

Commonsense170k uses Apache-2.0 License. ConvAI2 uses MIT license. Math10k uses Apache-2.0 License. We used AI assistants to help polish up the writing.