LoRTA: Efficient Low Rank Tensor Adaptation of Large Language Models

Anonymous ACL submission

Abstract

Low Rank Adaptation (LoRA) is a popular Parameter Efficient Fine Tuning (PEFT) method that effectively adapts large pre-trained models for downstream tasks. LoRA parameterizes model updates using low-rank matrices at each layer, significantly reducing the number of trainable parameters and, consequently, resource requirements during fine-tuning. However, the lower bound on the number of trainable parameters remains high due to the use of the low-rank matrix model. Recent works have addressed this limitation by proposing low rank tensor parameterizations for model updates. However, they only exploit redundancy across layers, or tensorize individual matrices using ad-hoc schemes that introduce additional hyperparameters. In this work, we propose a higher-order Candecomp/Parafac (CP) decomposition, enabling a more compact and flexible representation compared to existing matrix and tensor based PEFT methods. The proposed low rank tensor model can reduce the number of trainable parameters, while also allowing for finer-grained control over adapter size. Our experiments on Natural Language Understanding, Instruction Tuning, Preference Optimization and Protein Folding benchmarks demonstrate that our method can achieve a reduction in the number of parameters while maintaining comparable performance.

1 Introduction

The advent of Large Language Models (LLMs) – billion parameter scale models pre-trained on vast corpora of data – has enabled unprecedented capabilities across a wide range of tasks. However, as LLM sizes continue to grow exponentially, their computational and memory demands represent significant challenges, particularly for those lacking access to high-performance computing infrastructure (Varoquaux et al., 2024). This has spurred interest in parameter efficient fine tuning (PEFT) techniques (Ding et al., 2023), which facilitate the adaptation of LLMs to specific applications, downstream tasks or user preferences, by using only a small fraction of trainable parameters. Most importantly, they reduce GPU memory requirements, primarily by shrinking optimizer states (Liao et al., 2023). Moreover, they provide greater efficiency in storage and deployment, enabling the management of multiple fine-tuned LLMs with reduced storage footprints and faster load times (Sheng et al., 2023; Wen and Chaudhuri, 2024), which is particularly relevant for applications requiring rapid model switching across numerous task- or user-specific models.

Beyond computational benefits, PEFT techniques can also mitigate overfitting risks associated with fine-tuning high-capacity LLMs. By constraining model updates, PEFT methods can act as an implicit regularization mechanism, improving generalization (Fu et al., 2023; Sun et al., 2023). Parameter sharing, a well-established technique in deep learning architecture design, has been shown to improve generalization across various tasks such as protein folding (Jumper et al., 2021; Lin et al., 2023), image segmentation (Ronneberger et al., 2015), and generative modeling (Rombach et al., 2022). Incorporating parameter sharing in PEFT methods has also improved performance in specialized applications with limited data, such as in medical domains (Dutt et al., 2023; Zhu et al., 2024).

Low Rank Adaptation (LoRA) is a popular PEFT approach that uses a low rank parameterization of weight matrix updates (Hu et al., 2021). For instance, these allow to fine tune a 175 billion parameter LLM using only 5 million trainable parameters (Hu et al., 2021) without performance degradation. Since the model updates can be merged with the frozen weights, LoRA incurs no additional inference cost when deployed, unlike prompt (Li and Liang, 2021a; Liu et al., 2023) and adapter-based (Houlsby et al., 2019; He et al., 2021; Pfeiffer et al., 2020) PEFT methods.

However, the lower bound on trainable parameters often remains substantial for large-scale models. Recent works have aimed to further reduce the number of parameters in LoRA by using shared pseudorandom low rank projections (Zhang et al., 2023a; Kopiczko et al., 2023), or parameterizing low rank matrices using a pseudorandom basis (Koohpayegani et al., 2024). We show that the parameter-sharing schemes in these methods can be interpreted as low-rank tensor models with fixed random factors.

On the other hand, (Yang et al., 2024) leverage low rank tensor adapters by treating each weight update as a tensor with arbitrary dimensions. However, this ten-

sorization scheme not only introduces additional hyperparameters but also forfeits structural information and potential correlations among different weights. Other low-rank tensor adapter models recently proposed for vision transformers (Jie and Deng, 2023; Edalati et al., 2023) and LLMs (Bershatsky et al., 2024) treat model layers as an explicit mode, but do not exploit redundancies across attention matrices or heads. Moreover, they use tucker and tensor train models which are less parameter efficient and parsimonious than CANDECOMP/-PARAFAC (CP) models (Kolda and Bader, 2009).

Building on these insights, we propose LoRTA, a 5th-order CP-based low-rank factorization that unifies parameter updates across layers, heads, and attention matrices. To the best of our knowledge this is the first tensor based method to (i) exploit redundancy in weight updates across layers, heads, and attention matrices by representing updates as a unified 5th-order low-rank tensor (ii) employ a CP model.

We evaluate our method on diverse benchmarks including Natural Language Understanding, Instruction Tuning, Preference Optimization, and Protein Folding. Our experiments demonstrate that LoRTA can achieve up to an order of magnitude reduction in the number of trainable parameters compared to state-of-the-art PEFT methods, with minimal performance trade-offs.

2 Preliminaries

2.1 Transformer Architecture

We focus on the transformer architecture, although it can be naturally extended to other architectures such as Convolutional Neural Networks and Long Short Term Memory networks. We adopt the problem setting presented in (Thickstun, 2021). In the transformer model, an initial embedding layer maps input tokens to $d-$dimensional vector representations. These embeddings then pass through a series of layers, each performing multi-head attention, normalization and feed-forward operations. The input to the $l-$th layer of the transformer is a matrix $\boldsymbol{X}^{(l)} \in \mathbb{R}^{N\times d}$, where N is the number of queries, represented in a $d-$dimensional feature space. A vanilla transformer layer with H attention heads is then defined as follows:

$$\boldsymbol{X}^{(l+1)} = \texttt{LayerNorm}\left(\boldsymbol{Y}^{(l)} + \texttt{MLP}\left(\boldsymbol{Y}^{(l)}\right)\right)$$
$$\boldsymbol{Y}^{(l)} = \texttt{LayerNorm}\left(\boldsymbol{X}^{(l)} + \texttt{Attn}\left(\boldsymbol{X}^{(l)}\right)\right)$$
$$\texttt{Attn}\left(\boldsymbol{X}^{(l)}\right) = \boldsymbol{X}^{(l)} + \sum_{h=1}^{H} \texttt{softmax}\left(\frac{\boldsymbol{X}^{(l)}\boldsymbol{Q}_h^{(l)}\boldsymbol{K}_h^{(l)^T}\boldsymbol{X}^{(l)^T}}{\sqrt{d}}\right)\boldsymbol{X}^{(l)}\boldsymbol{V}_h^{(l)}\boldsymbol{P}_h^{(l)^T}$$
$$\texttt{MLP}\left(\boldsymbol{X}^{(l)}\right) = \texttt{ReLU}\left(\boldsymbol{X}^{(l)}\boldsymbol{G}_1^T + \boldsymbol{1}_N\boldsymbol{b}_1^T\right)\boldsymbol{G}_2^T + \boldsymbol{1}_N\boldsymbol{b}_2^T,$$

where $\boldsymbol{K}_h^{(l)}, \boldsymbol{Q}_h^{(l)}, \boldsymbol{V}_h^{(l)}, \boldsymbol{P}_h^{(l)} \in \mathbb{R}^{d\times d_H}$ are the key, query, value and projection matrices respectively, for head h and layer l.

2.2 Low Rank (matrix) Adaptation

LoRA modifies the pre-trained weights by adding a trainable update. Explicitly, at each layer and head h:

$$\boldsymbol{K}_h = \boldsymbol{K}_h^0 + d\boldsymbol{K}_h, \quad \boldsymbol{Q}_h = \boldsymbol{Q}_h^0 + d\boldsymbol{Q}_h,$$
$$\boldsymbol{V}_h = \boldsymbol{V}_h^0 + d\boldsymbol{V}_h, \quad \boldsymbol{P}_h = \boldsymbol{P}_h^0 + d\boldsymbol{P}_h,$$

where $\boldsymbol{K}^0, \boldsymbol{Q}^0, \boldsymbol{V}^0, \boldsymbol{P}^0$ denote the pre-trained weights and $d\boldsymbol{K}, d\boldsymbol{Q}, d\boldsymbol{V}, d\boldsymbol{P}$ the trainable adapters.

While each attention head's MLP contains two trainable matrices, $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$, our focus is on fine-tuning the attention matrices. This has been demonstrated to be effective for LLM adaptation (Hu et al., 2021; Kopiczko et al., 2023). Nevertheless, these methods can be easily extended to other parameters, including the MLP weights.

Let $\boldsymbol{W}_h \in \{\boldsymbol{Q}_h, \boldsymbol{K}_h, \boldsymbol{V}_h, \boldsymbol{P}_h\}$ for $h = 1, \ldots, H$ denote the query, key, value and projection matrices, respectively, for each attention head. After concatenating updates across all attention heads, we get:

$$d\tilde{\boldsymbol{W}} = (d\boldsymbol{W}_1, \ldots, d\boldsymbol{W}_H) \in \mathbb{R}^{d\times d}.$$

(Hu et al., 2021) proposed to parametrize the updates using rank-r matrices, which can be expressed as

$$d\tilde{\boldsymbol{W}} = \frac{\alpha}{r}\boldsymbol{A}\boldsymbol{B}^T, \quad \boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{d\times r}, \tag{1}$$

where α is a constant and r denotes the rank of the update. The scaling factor simply aims to reduce the efforts of re-tuning the learning rate when training adapters of varying rank. It has been shown that while this scaling heuristic works well for smaller ranks, it can be sub-optimal for larger ranks (Kalajdzievski, 2023). (Hayou et al., 2024) have also shown that setting the learning rate for the $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices appropriately can further improve convergence and performance.

Although LoRA is an efficient fine-tuning technique, the number of parameters required for each layer is at least $8 \cdot d \cdot r$. Thus, the total number of trainable parameters is:

$$\#\text{parameters (LoRA)} = 2 \cdot M \cdot d \cdot L \cdot r, \tag{2}$$

where L is the total number of layers and M the number of finetuned attention/projection matrices. Even with $r = 1$, this results in $4 \cdot M \cdot d \cdot L$ parameters. In practice, for LLMs with high dimensionality (d) and many layers (L), this lower bound can still lead to a significant number of trainable parameters.

LoRA has also been combined with model weight quantization (Dettmers et al., 2024), further decreasing resource requirements. Unlike adapter-based PEFT methods (Houlsby et al., 2019; Pfeiffer et al., 2020; Rücklé et al., 2020; He et al., 2021), LoRA does not introduce additional inference time overhead during deployment, as the trainable matrices can be integrated with the fixed weights.

Building upon this foundation, AdaLoRA (Zhang et al., 2023b) expands the LoRA technique by introducing dynamic rank adjustment for low-rank matrices

during fine-tuning. The fundamental concept involves optimally allocating the parameter resources by selectively pruning less crucial components of the matrices based on an importance metric. LoRA-FA (Zhang et al., 2023a) reduces the number of trainable parameters by freezing the $\boldsymbol{A}$ matrix to its random initialisation, while achieving similar performance to LoRA.

2.3 Tensor Algebra

In the following sections we introduce our proposed LoRTA framework, which is a tensor adaptation model for PEFT. To facilitate the upcoming analysis, we briefly present some tensor algebra preliminaries and refer the reader to Appendix A and (Sidiropoulos et al., 2017; Kolda and Bader, 2009) for further details.

A N-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is an N-way array indexed by $i_1, i_2, \ldots, i_N$ with elements $\mathcal{X}(i_1, i_2, \ldots, i_N)$. It consists of N types of modes: $\mathcal{X}(:, i_2, \ldots, i_N), \mathcal{X}(i_1, :, \ldots, i_N), \ldots, \mathcal{X}(i_1, i_2, \ldots, :)$. Any tensor can be decomposed as a sum of N-way outer products as: where $\boldsymbol{A_n} = [\boldsymbol{a}_n^1, \boldsymbol{a}_n^2, \ldots, \boldsymbol{a}_n^R] \in \mathbb{R}^{I_n \times R}$, $n = 1, \ldots, N$ are called the low rank factors of the tensor. The above expression represents the *canonical polyadic decomposition* (CPD) or *parallel factor analysis* (PARAFAC) (Harshman and Lundy, 1994) of a tensor. A tensor can be fully characterized by its latent factors, so we can represent a tensor by its CPD model as:

$$\mathcal{X} = [\![\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_N]\!] . \tag{3}$$

Unlike other tensor models, such as Tucker and Block Term Decomposition (BTD), the CPD model is unique under certain conditions. As a result, the CPD model is often preferred when the goal is to minimize the number of parameters.

A tensor can also be represented as a set of matrices, by fixing all the modes but two as:

$$\mathcal{X}[:, :, i_3, \ldots, i_N] = \tag{4}$$

$$\boldsymbol{A}_1 \left(\text{Diag}\left(\boldsymbol{A}_3\left(i_3, :\right)\right) \odot \cdots \odot \text{Diag}\left(\boldsymbol{A}_N\left(i_N, :\right)\right)\right) \boldsymbol{A}_2^T, \tag{5}$$

where $\text{Diag}\left(\boldsymbol{A}_n\left(i_n, :\right)\right)$ is the diagonal matrix with diagonal equal to $\boldsymbol{A}_N\left(i_n, :\right)$.

3 Low Rank Tensor adaptation

3.1 Parameter sharing across layers

To further increase the compression ratio in PEFT models, recent works (Kopiczko et al., 2023; Koohpayegani et al., 2024) suggest sharing parameters across layers that operate as predefined projection matrices. As we see next, this leads to tensor factorization models with fixed parameters.

Vector-based Random Matrix Adaptation (VeRA) (Kopiczko et al., 2023) have proposed to parameterize updates using two learnable vectors at each layer and fixed random matrices shared across all layers. The update at each layer can be expressed as

$$d\tilde{\boldsymbol{W}} = \boldsymbol{A}\text{Diag}\left(\boldsymbol{C}_D[l, :]\right) \boldsymbol{B}^T \text{Diag}\left(\boldsymbol{C}_B[l, :]\right), \tag{6}$$

where $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{d \times r}$ are the random projections, and $\boldsymbol{C}_D \in \mathbb{R}^{L \times r}$, $\boldsymbol{C}_B \in \mathbb{R}^{L \times d}$ are matrices that collect trainable vectors across layers. The model in 6 is a coupled matrix factorization model and is similar to a tensor model. In particular, if we remove the $\boldsymbol{C}_B$ term VeRA can be interpreted as a low-rank tensor CPD parameterization with fixed random factors. That is, the weight update $\tilde{\boldsymbol{W}}$ is a rank-r third order tensor $\mathcal{T} \in \mathbb{R}^{d \times d \times L}$. Note that, omitting the C_B term has been shown to lead to a small performance degradation unlike omitting C_D (Kopiczko et al., 2023).

Random Matrix basis Adaptation (NOLA) In a similar manner, (Koohpayegani et al., 2024) have proposed to parameterize the weight update by expressing the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ as linear combinations of fixed random basis matrices, that are shared across all layers. The weight update $d\boldsymbol{W}$ for layer l is then given by:

$$d\tilde{\boldsymbol{W}}_l = \sum_{i=1}^{k} \sum_{j=1}^{k} \alpha_{(i,l)} \beta_{(j,l)} \boldsymbol{A}_i \boldsymbol{B}_j^T, \tag{7}$$

where $\boldsymbol{A}_i, \boldsymbol{B}_j \in \mathbb{R}^{d \times r}$ are fixed random matrices, shared across all layers, and $\boldsymbol{\alpha}_l = \left\{\alpha_{(i,l)}\right\}_{i=1}^{K}$ and $\boldsymbol{\beta}_l = \left\{\beta_{(i,l)}\right\}_{i=1}^{K}$ are the learned coefficients for each layer. If we stack the random matrices $\boldsymbol{A}_i, \boldsymbol{B}_j \in \mathbb{R}^{d \times r}$ into tensors $\mathcal{A}$, $\mathcal{B}$ such that: $\mathcal{A}[:, :, i] = \boldsymbol{A}_i$ and $\mathcal{B}[:, :, j] = \boldsymbol{B}_j$, then 7 can be cast as:

$$\begin{aligned} d\tilde{\boldsymbol{W}}_l &= \sum_{i=1}^{k} \sum_{j=1}^{k} \alpha_{(i,l)} \beta_{(j,l)} \sum_{m=1}^{r} \mathcal{A}[:, m, i] \mathcal{B}[:, m, j]^T \\ &= \sum_{m=1}^{r} \mathcal{A}[:, m, :] \left(\boldsymbol{\alpha}_l \boldsymbol{\beta}_l^T\right) \mathcal{B}[:, m, :]^T \end{aligned}$$

and $d\tilde{\boldsymbol{W}}_l$ admits the following factorization. $d\tilde{\boldsymbol{W}}_l = \sum_{m=1}^{r} \boldsymbol{P}_A^{(m)} \left(\boldsymbol{\alpha}_l \boldsymbol{\beta}_l^T\right) \boldsymbol{P}_B^{(m)T}$, where $\boldsymbol{P}_A^{(m)} = \mathcal{A}[:, m, :]$, and $\boldsymbol{P}_B^{(m)} = \mathcal{B}[:, m, :]$ are also random projection matrices with different dimensions compared to $\boldsymbol{A}_i$, $\boldsymbol{B}_j$. As a result, NOLA can be viewed as the following tensor factorization model:

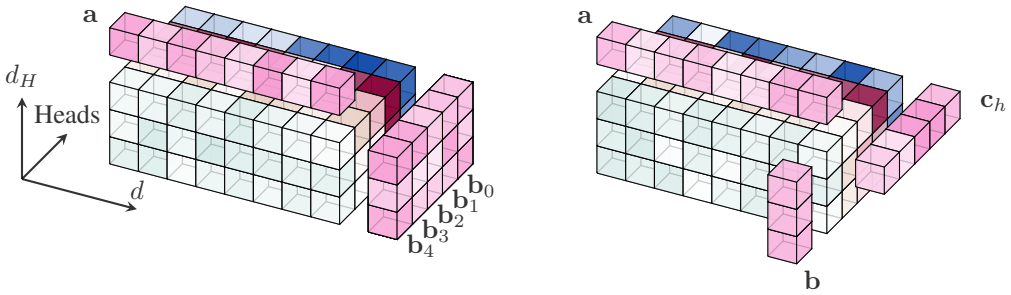
$$d\tilde{\mathcal{W}} = \sum_{m=1}^{r} [\![\boldsymbol{P}_A^{(m)} \tilde{\boldsymbol{A}}, \boldsymbol{P}_B^{(m)} \tilde{\boldsymbol{B}}, \boldsymbol{I}]\!] , \tag{8}$$

$$\tilde{\boldsymbol{A}}[:, l] = \boldsymbol{\alpha}_l, \tilde{\boldsymbol{B}}[:, l] = \boldsymbol{\beta}_l.$$

The expression in 8 is a a summation of CPD models, also known as Block Term Decomposition, which is an expressive tensor model, but can lack parsimony (Kolda and Bader, 2009).

3.2 LoRTA: A more efficient tensor model

In the previous section, we explored PEFT models that share parameters across layers, highlighting their correspondence to tensor factorization models. Namely, VeRA and NOLA utilize fixed projection matrices shared across layers. However, this strategy can result in models that are larger than necessary relative to



LoRA. LoRTA.

Figure 1. Illustration of a rank 1 adapter for a single weight matrix with multiple heads. (Left) The LoRA update for head h is computed as $d\boldsymbol{W}_h = \boldsymbol{b}_h \circ \boldsymbol{a}$. (Right) The update using a third order low rank tensor model is computed as $dW_h = \boldsymbol{b} \circ \boldsymbol{c}[h] \circ \boldsymbol{a}$. Both models have the same tensor rank, but the latter has less parameters.

their degrees of freedom due to the inclusion of these random matrices. Although these matrices can be generated on the fly by solely storing the pseudo-random number generator seed, this still incurs additional resource demands during training, and increases loading time for inference.

To address this issue, we propose modeling the trainable adapters using a low-rank CPD structure. This choice is motivated by the favorable properties of CPD: it is universal, capable of exactly factorizing any tensor, yet remains concise and parsimonious, typically requiring only a small number of parameters to achieve low approximation error (Sidiropoulos et al., 2017). This contrasts with tensor adapters used in vision (Jie and Deng, 2023) and recently in LLM finetuning (Bershatsky et al., 2024), which rely on Tucker and Tensor-Train models. In fact, for small ranks, CPD is equivalent to Tucker when the core tensor in Tucker is the identity tensor. However Tucker is always parametrized with a dense tensor and therefore requires a larger number of parameters for the same rank.

LoRTA represents all weight updates as a 5th-order tensor $d\tilde{\mathcal{W}} \in \mathbb{R}^{d \times \frac{d}{H} \times H \times L \times M}$. By integrating updates of layers, heads and the $\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$, $\boldsymbol{P}$ matrices into a unified low-rank CPD tensor model, LoRTA exploits redundancy across different modes of the tensor. This approach can thus not only improve parameter efficiency but also facilitate learning by exploiting the shared information among various components of the model. This contrasts with existing PEFT approaches, which tensorize each weight update independently (Yang et al., 2024) or only share parameters across layers (Jie and Deng, 2023; Bershatsky et al., 2024). In order to illustrate how additional tensor modes can result in parameter efficiency gains, Figure 1 compares – for a single weight update – LoRA with a rank one tensor model that adds attention heads as a mode.

By utilizing a low-rank CPD model, we can express this tensor as:

$$d\tilde{\mathcal{W}} = [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}_H, \boldsymbol{C}_L, \boldsymbol{C}_M]\!],$$

where $\boldsymbol{A} \in \mathbb{R}^{d \times r}$ and $\boldsymbol{B} \in \mathbb{R}^{\frac{d}{H} \times r}$ are factor matrices for the input and output dimensions, respectively, and $\boldsymbol{C}_H \in \mathbb{R}^{H \times r}$, $\boldsymbol{C}_L \in \mathbb{R}^{L \times r}$, $\boldsymbol{C}_M \in \mathbb{R}^{4 \times r}$ are factor matrices for the attention heads, layers, and the four matrices Q, K, V, P. Each weight matrix update can then be retrieved as:

$$d\tilde{\mathcal{W}}[:,:,k,l,i] = \boldsymbol{A}\left(\mathrm{Diag}\left(\boldsymbol{C}_H[k,:]\right)\mathrm{Diag}\left(\boldsymbol{C}_L[l,:]\right)\mathrm{Diag}\left(\boldsymbol{C}_M[i,:]\right)\right)\boldsymbol{B}^\top,$$

where k indexes the attention heads, l indexes the layers, and i indexes the matrices $\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$, $\boldsymbol{P}$. Note that, unlike previous implicit tensor models such as NOLA

Method	Update Tensor shape	Tensor Model	Parameters	r=4	r=64
LoRA	$ML \times d \times d$	Matrix-Batch	$2MLdr$	2.1M	33M
LoReTTA	$ML \times d \times d$	Custom	$2MLr^2 \sum_i k_i$	92k	50M
LoTR	$ML \times d \times d$	Tucker2	$M(Lr^2 + 2dr)$	33k	786k
FacT-TT	$ML \times d \times d$	Tensor-Train	$MLr^2 + 2dr$	33k	786k
FacT-TK	$ML \times d \times d$	Tucker3	$(2d + ML)r + r^3$	33k	790k
Ours	$M \times L \times d \times d/h \times h$	CP	$(d + d/h + h + L + M)r$	17k	274k

Table 1: Number of parameters of different low rank PEFT methods as a function of the number of finetuned attention/projection matrices M, the number of layers, L, the embedding dimension d, the number of heads h and the tensor rank of the update, r. For LoreTTA, k_i are hyperparameters that must satisfy $\prod_i k_i = dr$ and $k_i \geq r$ for all i. We also include the number of parameters for the Llama2-7b architecture when finetuning only M=2 attention matrices (e.g. Q and V) for different ranks. For LoreTTa we use $k_1 = \ldots = k_6 = 5$ for $r = 4$ and $k_1 = k_2 = k_3 = 64$ for $r = 64$.

and VeRA, which rely on fixed random projections or parameters and learn only scaling coefficients, our proposed model is *fully trainable*. All factor matrices ($\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}_H$, $\boldsymbol{C}_L$, $\boldsymbol{C}_M$) are learned during training, providing greater expressiveness and forgoing the dependency on pre-defined random bases or projections.

Table 1 shows how the CP low rank tensor parameterization leads to better scaling in the number of parameters with respect to the tensor rank r. Moreover, our higher-order weight update tensorization improves scaling in terms of transformer architecture hyperparameters, namely the embedding dimension d, number of attention heads H, and number of fine-tuned attention matrices M.

3.3 Other Low Rank Tensor models in PEFT

As mentioned in the previous section, existing PEFT tensor-based models differ from our method both in their parameter-sharing schemes, which result from different weight update tensorization approaches, as well as in the low-rank tensor models they employ. Below, we provide a concise overview of these approaches which intends to highlight the differences with LoRTA; further details are available in Appendix C and the provided references.

FaCT & LoTR In the context of vision transformers, (Jie and Deng, 2023) have proposed to represent updates across all layers as a third order tensor $d\tilde{\mathcal{W}} \in \mathbb{R}^{L \times d \times d}$. They propose two parameterizations of $d\tilde{\mathcal{W}}$, namely, a Tensor Train and Tucker3 low rank tensor models. Recently, (Bershatsky et al., 2024) have proposed to apply the same tensorization across layers to fine-tune LLMs, but using a low rank Tucker2 tensor model to parameterize updates:

$$d\tilde{\mathcal{W}} =; ; \boldsymbol{G} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \tag{9}$$

where $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{d \times r}$ and $\boldsymbol{G} \in \mathbb{R}^{L \times r \times r}$.

LoreTTA (Yang et al., 2024) propose two methods that employ low rank tensor models. However, these models do not share parameters across layers, they reparameterize low rank matrix adapters using low rank tensor models. In LoreTTA-rep a low rank matrix model is first applied to each weight update in the same manner as described for LoRA in Equation (17). Then each of the ML resulting LoRA factors $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{d \times r}$ are expressed as a n-th order tensor with arbitrary dimensions, i.e. $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{k_1 \times \ldots \times k_N}$. Finally, each of these tensors is parametrized Tensor Train model, explicitly,

$$\mathcal{A} = \prod_{i=1} \boldsymbol{G}_i \quad \text{where} \quad \boldsymbol{G}_i \in \mathbb{R}^{r \times k_i \times r}. \tag{10}$$

We highlight that the added dimensions k_i are hyperparameters that must satisfy $\prod_i k_i = dr$ and $k_i \geq r$ for all i; otherwise, it would induce a new tensor rank deficiency. Moreover, choosing appropriate values of k_i might be challenging and necesitate further hyperparameter tuning. (Yang et al., 2024) also proposed LoReTTA-adp, applying a tucker parameterization to an adapter method, which unlike our method and the rest of the aforementioned methods adds new parameters to the model and thus can not be merged into the original weights, thereby incurring additional inference costs.

4 Experiments

4.1 Natural Language Understanding

We evaluate our approach by fine-tuning RoBERTa models on the General Language Understanding Evaluation (GLUE) (Wang et al., 2018) benchmark. We conduct experiments across three distinct settings previously reported in the literature by (Bershatsky et al., 2024), (Yang et al., 2024) and (Kopiczko et al., 2023). These settings differ in hyperparameters, including the number of training epochs, different learning rates for the classification head and encoder, the learning rate decay strategy (linear vs fixed), the use of different scaling parameters α, and the grid search ranges. Because the best results on the validation set across a grid of hyperparameter values are reported, performance for the same baseline method can vary considerably across settings (see, for example, LoRA performance reported by (Hu et al., 2021), (Yang et al., 2024) and (Bershatsky et al., 2024)). Therefore, we provide an evaluation of our method in a variety of experimental conditions, while also maintaining the original configurations in which state-of-the-art methods were originally reported. Detailed settings can be found in Appendix E.1.

We also finetuned Llama2 models (Touvron et al., 2023) on question-answering (QA) tasks SQuAD (Rajpurkar et al., 2016), DROP (Dua et al., 2019), COPA (Roemmele et al., 2011), and ReCoRD (Zhang et al., 2018), following the experimental setting outlined by (Yang et al., 2024). For these tasks, we used a randomly selected subset of 1,000 samples to simulate a low-data regime and increase the task difficulty. All classification tasks are tackled as language modeling tasks following the prompt-based fine-tuning approach described by (Malladi et al., 2023).

Baselines We benchmark our method against the following methods:

- **Full finetuning**: all parameters are trained.
- **IA3** (Liu et al., 2022): rescales activations with learned vectors
- **Prefix** (Li and Liang, 2021b): prepends learnable continuous vectors (prefixes) to the input embeddings.
- **LoRA** (Hu et al., 2021), **LoRA-FA** (Zhang et al., 2023a) and **VeRA** (Kopiczko et al., 2023), **LoTR** (Bershatsky et al., 2024), **LoReTTA-rep** (Yang et al., 2024): As previously described.
- We omit **Adapter$^{\mathbf{H}}$** (Houlsby et al., 2019), **Adapter$^{\mathbf{P}}$** (Pfeiffer et al., 2020), **Bitfit** (Zaken et al., 2021), **AdapterDrop** (Rücklé et al., 2020), and other methods that are customarily reported

	Method	# Trainable Parameters	SST-2	MRPC	CoLA	QNLI	RTE	STS-B	Avg.
LoReTTA	LoRA (r=8)	630k	94.01	91.48	62.08	92.39	74.51	84.69	83.19
	LoReTTA rep	70k	94.28	90.63	61.72	92.40	74.42	89.24	83.78
	LoRTA (r=20)	48k	94.27	92.04	63.35	91.48	75.09	89.82	**84.34**
	LoRTA (r=12)	29k	93.81	91.13	61.40	92.04	74.73	89.64	83.79
LoTR	LoRA (r=8)	300k	94.2	88.0	61.1	91.3	73.0	90.7	83.05
	LoTR	74k	93.0	85.9	60.5	90.0	66.0	91.9	81.22
	LoRTA (r=16)	15k	94.73	90.44	64.32	92.37	76.9	90.25	**84.84**
	LoRTA (r=4)	**3.4k**	94.61	89.21	60.55	90.61	76.9	89.97	83.6
VeRA	LoRA	800k	96.2	90.2	68.2	**94.8**	85.2	**92.3**	**87.8**
	LoRA-FA	3.7M	96.0	90.0	68.0	94.4	86.1	92.0	87.7
	VeRA	61k	96.1	**90.9**	68.0	94.4	**85.9**	91.7	**87.8**
	LoRTA (r=8)	**9k**	96.3	89.5	65.1	94.3	85.6	91.1	85.7

Table 2: Performance of RoBERTa Base and Large models on GLUE tasks under three different experimental settings reported by LoReTTA (Yang et al., 2024), LoTR (Bershatsky et al., 2024), and VeRA (Kopiczko et al., 2023). In LoReTTA, LoRTA is applied to the encoder and LoRA to the classifier with the same rank, while for LoTR and VeRA, LoRTA is applied only to the encoder. Trainable parameters include the classifier for LoReTTA but exclude it for LoTR and VeRA, where it is fully trained. VeRA results use RoBERTa Large, whereas LoTR and LoReTTA use RoBERTa Base.

in PEFT literature but have been outperformed by more recent methods in these settings.

The results in Table 2 show that LoRTA can achieve comparable or slightly superior performance with less trainable parameters compared to state of the art tensor based PEFT methods LoreTTA (Yang et al., 2024) and LoTR (Bershatsky et al., 2024) when finetuning RoBERTA base on GLUE tasks. Similarly, for RoBERTa large LoRTA can also achieve a 6x reduction in the number of trainable parameters with only small drop in average performance (2%) when compared to (Kopiczko et al., 2023). In this settings we did not tune the hyperparameters for our method as extensively as baselines, and thus this gap could be further reduced.

In Llama QA experiments, shown in Table 3, full fine-tuning (FT) achieves the highest average score (77.3) with 7 billion trainable parameters, but among the PEFT methods LoRTA (r=8) achieves the highest average score (76.7) with just 0.03 million parameters, representing a 17x reduction in parameter count with respect to the most efficient method.

4.2 Instruction Tuning

We fine-tune the 7 billion parameter Llama2 (Touvron et al., 2023) models on the cleaned Alpaca instruction tuning dataset (Taori et al., 2023). While more recent models and tasks exist, we select this well-studied setting because it enables direct comparison with an extensive body of prior work, and maintains methodological consistency with our earlier experiments on NLU tasks. We train for one epoch, preceded by a warm-up learning rate sweep as in the standard setting. Other hyperparameters are detailed in Appendix E.2.

As shown in Figure 2, LoRTA effectively reduces the number of parameters to a fraction of those required by the lowest rank in LoRA, with only a small performance cost. In this setting the validation cross entropy decreases monotonically with the number of parameters used, both in training and testing, and LoRTA even demonstrates superior performance with fewer parameters for ranks 96 and 192.

Although cross entropy (and perplexity) has been shown to be correlated with diverse downstream performance metrics (Dubois et al., 2024), we provide additional evaluations using other standard LLM-as-a-judge benchmarks in Appendix F, which also show LoRTA attains comparable performance to LoRA at a fraction of parameters.

4.3 Preference Optimization

Among the various existing techniques to align LLMs with human preferences on specific tasks–see, for example, (Kaufmann et al., 2023) and references therein– we utilize Direct Preference Optimization (DPO) (Rafailov et al., 2024) due to its widespread use. We set the regularization coefficient that penalizes deviations from the pre-trained model's outputs (β) to 0.1. We use the cleaned version of the Intel Orca dpo pairs dataset[1]. We use Huggingface Transformer Reinforcement Learning (trl) library[2]. Consistent with our previous experiments, we use Llama2-7b as our base model. For a complete description of hyperparameters see Appendix E.3.

The results in Table 4 compare both methods using rank 1 updates. LoRTA achieves comprable performance to LoRA in terms of validation loss, while using only 4k parameters—a 99% reduction from LoRA's 524k parameters. In addition, LoRTA attained a slight

[1] https://huggingface.co/datasets/argilla/distilabel-intel-orca-dpo-pairs

[2] https://github.com/huggingface/trl

Method	# Trainable Parameters	COPA	ReCoRD	SQuAD	DROP	Avg.
Full	7B	86	81.1	90.71	51.38	77.3
LoRA (r=8)	4.19M	81	79.4	90.56	45.96	74.2
Prefix	1.31M	83	81.0	90.56	45.95	75.1
IA3	0.60M	80	81.5	89.41	39.37	72.6
LoRETTA rep	0.51M	86	80.3	88.47	42.71	74.4
LoRTA (r=4)	**0.02M**	87	81.1	87.4	44.04	74.9
LoRTA (r=8)	**0.03M**	87	81.6	88.5	49.7	**76.7**

Table 3: Llama2-7B performance on SuperGLUE and question-answering tasks (SQuAD, DROP). We follow the experimental setup used by (Yang et al., 2024).

	# Parameters	Val. Loss	MT-bench Score
LoRA	524 k	0.44	4.08
LoRTA	4 k	0.43	4.14

Table 4: Llama2-7b fineuned using DPO on the orca dataset. Both methods use rank 1. We report the DPO loss for held-out data (lower is better) and the average score across MT-bench tasks.

improvement in MT-Bench (Zheng et al., 2023) performance, showing differences in generalization across tasks. Additional results for different ranks can be found on Appendix F. Unlike instruction tuning, preference across ranks exhibits non-monotonic behaviour, and larger ranks do not lead to performance improvements.

4.4 Protein Folding

Protein folding, the process by which a protein's amino acid sequence determines its three-dimensional structure, is a fundamental problem in molecular biology. Accurate prediction of protein structures from their sequences has significant implications for understanding protein function and designing new proteins for therapeutic purposes. ESMFold (Lin et al., 2023) is a frontier model for this task trained in two stages. First, ESM-2, a BERT-based (Devlin et al., 2019) protein language model, is trained with the masked-language-modeling objective on amino acid sequences. This unsupervised pretraining allows the model to capture complex patterns and relationships within protein sequences. Remarkably, valuable structural information emerges in the model's features during this process (Rao et al., 2020). In the second stage, ESM-2 is frozen, and a model head predicting three-dimensional protein structures is trained on top of language model features.

We re-train ESMFold in the second stage – fine-tuning ESM-2 parameters (we use ESM-2 35M instead of the ESM-2 3B model used in (Lin et al., 2023) due to compute constraints) with LoRA and LoRTA instead of freezing them. We evaluate performance with the Local Distance Difference Test for Cα atoms (LDDT-Cα) (Mariani et al., 2013) – that measures accuracy of predicted protein structures by comparing the distance between alpha carbons in predicted and true structures. LDDT-Cα ranges from 0 (poor accuracy) to 1 (perfect match). See Appendix E.4 for experiment details.

As shown in Table 14, LoRTA with ranks 1 and 8 achieves performance comparable to LoRA rank 1, despite using an order of magnitude fewer parameters. In Appendix F, we report training losses and results for

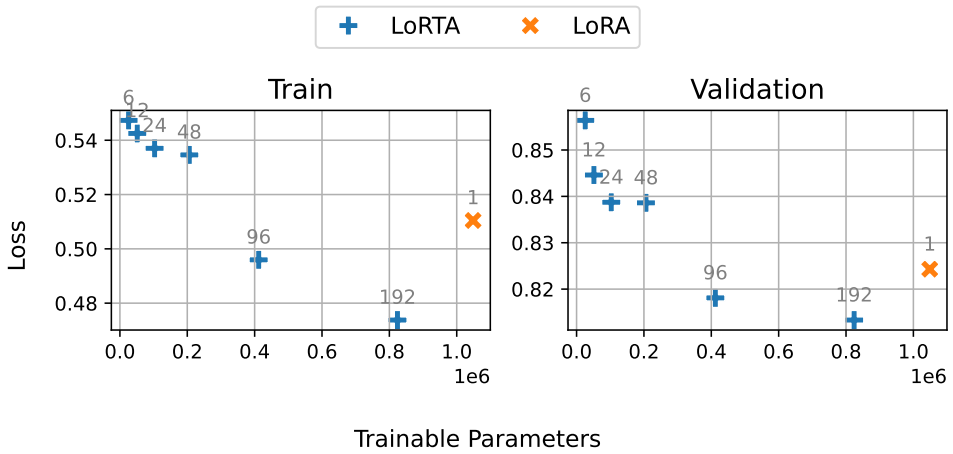


Figure 2. Mean cross-entropy loss on training and testing data for Llama2-7b on the Alpaca dataset vs number of trainable parameters for different adapter ranks. Lower is better. Numbers on top of markers denote the adapter rank.

	Rank	# Trainable Parameters ($\times 10^4$)	LDDT-Ca
LoRA	1	3.5	0.67
LoRTA	1	0.1	0.66
	8	0.5	0.67

Table 5: Mean LDDT-Cα on held-out data. Higher is better. LoRTA rank 1 is competitive with LoRA rank 1 on the test set despite having 64x fewer parameters.

larger ranks. Notably, larger ranks for LoRTA do not improve performance.

4.5 Computational Advantages

Reduced I/O times for concurrent adapters. During training, the GPU memory usage and training times of most low rank methods are comparable. Nonetheless, further reductions in adapter size are motivated by (i) lower storage requirements (ii) the need to improve task-switching efficiency and minimize storage requirements in scenarios involving a large number—potentially thousands—of adapters. Frequent CPU-GPU transfers for loading adapters in such settings can introduce significant overhead. By further compressing parameters, it becomes feasible to load several customized models with a shared base LLM in GPU memory, substantially enhancing scalability and performance in multi-task environments.

r	n	Transfer time (ms)		
		Lora	Lotr	Lorta
4	1	10.0 (9.0)	0.4 (0.2)	**0.12** (0.03)
	10	12.64 (0.08)	3.25 (0.05)	**1.06** (0.01)
	100	144.0 (2.0)	17.1 (0.2)	**5.5** (0.2)
64	1	23.0 (5.0)	3.4 (0.2)	**1.08** (0.01)
	10	216.0 (2.0)	41.02 (0.04)	**8.811** (0.005)
	100	2272 (31)	375.1 (0.2)	**61.5** (0.1)

Table 6: CPU to GPU transfer time in milliseconds (mean (std) across 20 repetitions) of N concurrent adapters with rank r for Llama2-7b using an NVIDIA A6000. Boldface indicates smallest value within one standard deviation.

To illustrate the potential latency reduction obtained from more compact adapters, in Table 6 we present CPU to GPU transfer times for different adapter methods with the same rank. Additional hardware profiling results for memory usage and training time can be found on Appendix G.

Tensor Decomposition of trained adapters is challenging. We also explore obtaining similarly compact representations by decomposing pre-trained LoRA adapters. To empirically demonstrate the challenges of post-training tensor decomposition, we conducted experiments using trained LoRA adapters from our preference optimization task (using Llama2-7b as a base model, and rank 8 for the adapters). Using TensorLy's[3] implementation of CP decomposition via alternating least squares (ALS), we attempted to decompose each weight matrix $W \in \mathbb{R}^{d\times d}$ into a rank-8 CP model by reshaping it into a 3rd-order tensor $\mathcal{T} \in \mathbb{R}^{d\times \frac{d}{h}\times h}$, where h is the number of attention heads. This tensorization scheme limits our proposed LoRTA model's decomposition to attention heads, i.e., without the additional structure across layers and matrices.

Metric	Mean	Median	Max	Std
Relative Error	0.827	0.838	0.910	0.053
R^2	0.312	0.297	0.506	0.086

Table 7: Approximation quality metrics for CP decomposition of trained LoRA weights across all layers and Q/V matrices. The high relative error and low R^2 scores indicate poor reconstruction quality.

The results in Table 7 reveal remarkably high approximation errors that render the compressed adapters ineffective, even when targeting the same parameter count achieved by direct tensor-based training. This suggests incorporating low rank tensor structure during training can guide the optimization toward fundamentally different, more compressible parameter updates. This finding parallels similar observations in neural network compression (Hoefler et al., 2021), where incorporating structural constraints during training often yields better results than post-hoc pruning.

5 Conclusion

We have introduced LoRTA, a novel approach that employs a low-rank tensor model for LLM updates. By extending low-rank adaptation to higher-order tensors, LoRTA overcomes the inherent lower bounds on the number of trainable parameters while offering finer-grained control over adapter size. Our experiments across various benchmarks demonstrate that LoRTA achieves comparable and sometimes superior performance than baselines at a reduced parameter count.

6 Limitations and Future Work

While our experiments demonstrate the LoRTA can be used to finetune models effiently across various settings comprising different tasks and model architectures, there are several important limitations and directions for future empirical research on the proposed adaptation method.

First, we have shown that previous works have implicitly utilized low-rank tensor models with random factors. Nothing precludes our higher-order tensor model from using randomized factors for increased efficiency—a potential direction for future work that could further reduce computational overhead. Lastly, developing more

[3] https://tensorly.org

efficient implementations of tensor operations that result in greater memory efficiency also remains a relevant future work direction which could make LoRTA even more suitable for resource-constrained environments.

Second, our evaluation was constrained to models up to 7B parameters due to computational limitations, with LLaMA 2 being the latest model tested. Further research is needed to assess LoRTA's scalability and effectiveness on larger models (e.g., 70B+ parameters) and more recent architectures. Additionally, understanding how parameter efficiency gains evolve with increasing model size remains an open question. Expanding our evaluation beyond standard benchmarks to multimodal models, text-to-speech systems, and domain-specific adaptation scenarios could provide deeper insights into the method's generalizability and robustness. Moreover, our study did not incorporate human evaluations, which could offer more nuanced assessments of LoRTA's impact on model quality and usability.

While our method shows promise for concurrent adapter scenarios, further research is needed to evaluate its effectiveness in these settings, including adapter composition, cross-task transfer, and adapter merging. Additionally, exploring LoRTA in the context of Mixture of Experts (MoE) architectures—where experts could be parameterized as tensor factors—represents an interesting direction that could enhance both parameter and computational efficiency. The potential for sharing tensor factors across experts or dynamically adjusting tensor ranks based on task complexity remains unexplored.

The current tensorization scheme, while effective, represents just one possible approach. Alternative schemes might offer different efficiency-performance trade-offs or be better suited for specific architectures or tasks. For instance, incorporating additional modes based on model-specific features (like relative position embeddings or sliding window attention patterns) could potentially yield further improvements. Moreover, our method currently focuses on attention matrix adaptation, and extending it to other components like MLPs or embeddings warrants investigation.

Further empirical investigation could provide valuable insights through ablation studies on the impact of tensor rank across different settings, detailed analysis of the learned tensor factors, and examination of how different tensorization schemes affect various aspects of model behavior.

Our work addresses only the parameter-efficient finetuning aspect of model adaptation. Future research could explore combining LoRTA with other efficiency techniques such as quantization, pruning, or activation compression. Additionally, while we demonstrated improved I/O efficiency for concurrent adapters, developing more efficient implementations of tensor operations could further reduce memory usage and training time. This includes leveraging hardware-specific optimizations and exploring methods to compress or efficiently compute intermediate activations.

From a theoretical perspective, several questions remain open. Understanding why tensor-based adapters provide an effective inductive bias for model adaptation, and characterizing what different adapter architectures learn, could provide insights for designing better adaptation methods. Additionally, while we focused on CP decomposition due to its parameter efficiency, comparative studies with other tensor decompositions (e.g., Tucker, Tensor Train) could reveal interesting trade-offs between expressivity and efficiency. Finally, while our preliminary experiments suggest that incorporating low-rank structure during training leads to more compressible updates than post-hoc decomposition, a deeper understanding of this phenomenon could inform the development of improved adaptation methods.

References

Gustaf Ahdritz, Nazim Bouatta, Christina Floristean, Sachin Kadyan, Qinghui Xia, William Gerecke, Timothy J. O'Donnell, Daniel Berenberg, Ian Fisk, Niccolò Zanichelli, Bo Zhang, Arkadiusz Nowaczynski, Bei Wang, Marta M. Stepniewska-Dziubinska, Shang Zhang, Adegoke Ojewole, Murat Efe Guney, Stella Biderman, Andrew M. Watkins, Stephen Ra, Pablo Ribalta Lorenzo, Lucas Nivon, Brian Weitzner, Yih-En Andrew Ban, Shiyang Chen, Minjia Zhang, Conglong Li, Shuaiwen Leon Song, Yuxiong He, Peter K. Sorger, Emad Mostaque, Zhao Zhang, Richard Bonneau, and Mohammed AlQuraishi. 2024. Openfold: retraining alphafold2 yields new insights into its learning mechanisms and capacity for generalization. Nature Methods, 21(8):1514–1524.

Nadav Benedek and Lior Wolf. 2024. Prilora: Pruned and rank-increasing low-rank adaptation. In Findings of the Association for Computational Linguistics: EACL 2024, pages 222–234. Association for Computational Linguistics.

Daniel Bershatsky, Daria Cherniuk, Talgat Daulbaev, Aleksandr Mikhalev, and Ivan Oseledets. 2024. Lotr: Low tensor rank weight adaptation. arXiv preprint arXiv:2402.01376.

Eric L. Buehler and Markus J. Buehler. 2024. X-lora: Mixture of low-rank adapter experts, a flexible framework for large language models with applications in protein mechanics and molecular design. APL Machine Learning, 2(2):026119.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. Advances in Neural Information Processing Systems, 36.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. Preprint, arXiv:1810.04805.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen,

Chi-Min Chan, Weize Chen, et al. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. Nature Machine Intelligence, 5(3):220–235.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P Bosma, Zongwei Zhou, Tao Wang, Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. 2022. GLaM: Efficient scaling of language models with mixture-of-experts. In Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pages 5547–5569. PMLR.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. 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 2368–2378. Association for Computational Linguistics.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. Preprint, arXiv:2404.04475.

Raman Dutt, Linus Ericsson, Pedro Sanchez, Sotirios A Tsaftaris, and Timothy Hospedales. 2023. Parameter-efficient fine-tuning for medical image analysis: The missed opportunity. arXiv preprint arXiv:2305.08252.

Ali Edalati, Marawan Gamal Abdel Hameed, and Ali Mosleh. 2023. Generalized kronecker-based adapters for parameter-efficient fine-tuning of vision transformers. In 2023 20th Conference on Robots and Vision (CRV), pages 97–104.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. Journal of Machine Learning Research, 23(120):1–39.

Zihao Fu, Haoran Yang, Anthony Man-Cho So, Wai Lam, Lidong Bing, and Nigel Collier. 2023. On the effectiveness of parameter-efficient fine-tuning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 37, pages 12799–12807.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752.

Seokil Ham, Jinyoung Lee, and Minsoo Park. 2024. Parameter efficient mamba tuning via projector-targeted diagonal-centric linear transformation (prodial). arXiv preprint arXiv:2411.15224.

Richard A Harshman and Margaret E Lundy. 1994. Parafac: Parallel factor analysis. Computational Statistics & Data Analysis, 18(1):39–72.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. Lora+: Efficient low rank adaptation of large models. arXiv preprint arXiv:2402.12354.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. arXiv preprint arXiv:2110.04366.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. 2021. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. Journal of Machine Learning Research, 22(241):1–124.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In International conference on machine learning, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685.

Shibo Jie and Zhi-Hong Deng. 2023. Fact: Factor-tuning for lightweight adaptation on vision transformer. Proceedings of the AAAI Conference on Artificial Intelligence, 37(1):1060–1068.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. 2021. Highly accurate protein structure prediction with alphafold. Nature, 596(7873):583–589.

Damjan Kalajdzievski. 2023. A rank stabilization scaling factor for fine-tuning with lora. arXiv preprint arXiv:2312.03732.

Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. 2023. A survey of reinforcement learning from human feedback. arXiv preprint arXiv:2312.14925.

Hyunwoo Kim, Wei Zhang, Anika Patel, et al. 2025. Mambapeft: Exploring parameter-efficient fine-tuning for mamba. In International Conference on Learning Representations (ICLR) 2025. OpenReview.

Tamara G Kolda and Brett W Bader. 2009. Tensor decompositions and applications. SIAM review, 51(3):455–500.

Soroush Abbasi Koohpayegani, KL Navaneet, Parsa Nooralinejad, Soheil Kolouri, and Hamed Pirsiavash. 2024. Nola: Compressing lora using linear combination of random basis. In The Twelfth International Conference on Learning Representations.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki Markus Asano. 2023. Vera: Vector-based random matrix adaptation. arXiv preprint arXiv:2310.11454.

Xiang Lisa Li and Percy Liang. 2021a. Prefix-tuning: Optimizing continuous prompts for generation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 4582–4597.

Xiang Lisa Li and Percy Liang. 2021b. Prefix-tuning: Optimizing continuous prompts for generation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 4582–4597.

Baohao Liao, Shaomu Tan, and Christof Monz. 2023. Make pre-trained model reversible: From parameter to memory efficient fine-tuning. In Neural Information Processing Systems.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024a. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. Proceedings of Machine Learning and Systems, 6:87–100.

Xinyu Lin, Jinpeng Wang, Yong Zhang, Jingbo Zhang, and Ji-Rong Wen. 2024b. Data-efficient fine-tuning for llm-based recommendation. arXiv preprint arXiv:2401.17197.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. 2023. Evolutionary-scale prediction of atomic-level protein structure with a language model. Science, 379(6637):1123–1130.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. Advances in Neural Information Processing Systems, 35:1950–1965.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2023. Gpt understands, too. AI Open.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. Advances in neural information processing systems, 36:21702–21720.

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D. Lee, Danqi Chen, and Sanjeev Arora. 2023. Fine-tuning language models with just forward passes. arXiv preprint arXiv:2305.17333.

Valerio Mariani, Marco Biasini, Alessandro Barbato, and Torsten Schwede. 2013. lddt: a local superposition-free score for comparing protein structures and models using distance difference tests. Bioinformatics, 29(21):2722–2728.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, et al. 2023. Rwkv: Reinventing rnns for the transformer era. arXiv preprint arXiv:2305.13048.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2020. Adapterfusion: Non-destructive task composition for transfer learning. arXiv preprint arXiv:2005.00247.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. Advances in Neural Information Processing Systems, 36.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2383–2392. Association for Computational Linguistics.

Roshan Rao, Joshua Meier, Tom Sercu, Sergey Ovchinnikov, and Alexander Rives. 2020. Transformer protein language models are unsupervised structure learners. bioRxiv.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In Proceedings of the 2011 AAAI Spring Symposium Series. Association for the Advancement of Artificial Intelligence.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bj"orn Ommer. 2022. High-resolution image synthesis with latent diffusion models. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10684–10695.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention, pages 234–241. Springer.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna

Gurevych. 2020. Adapterdrop: On the efficiency of adapters in transformers. arXiv preprint arXiv:2010.11918.

Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. 2013. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In 2013 IEEE international conference on acoustics, speech and signal processing, pages 6655–6659. IEEE.

Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, et al. 2023. S-lora: Serving thousands of concurrent lora adapters. arXiv preprint arXiv:2311.03285.

Nicholas D Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E Papalexakis, and Christos Faloutsos. 2017. Tensor decomposition for signal processing and machine learning. IEEE Transactions on Signal Processing, 65(13):3551–3582.

Pablo Sprechmann, Alexander M Bronstein, and Guillermo Sapiro. 2015. Learning efficient sparse and low rank models. IEEE transactions on pattern analysis and machine intelligence, 37(9):1821–1833.

Yang Sui, Miao Yin, Yu Gong, Jinqi Xiao, Huy Phan, and Bo Yuan. 2024. Elrt: Efficient low-rank training for compact convolutional neural networks. arXiv preprint arXiv:2401.10341.

Simeng Sun, Dhawal Gupta, and Mohit Iyyer. 2023. Exploring the impact of low-rank adaptation on the performance, efficiency, and regularization of rlhf. arXiv preprint arXiv:2309.09055.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

John Thickstun. 2021. The transformer model in equations. University of Washington: Seattle, WA, USA.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288.

Gaël Varoquaux, Alexandra Sasha Luccioni, and Meredith Whittaker. 2024. Hype, sustainability, and the price of the bigger-is-better paradigm in ai. Preprint, arXiv:2409.14160.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. arXiv preprint arXiv:1804.07461.

Hongyi Wang, Saurabh Agarwal, Yoshiki Tanaka, Eric Xing, Dimitris Papailiopoulos, et al. 2023. Cuttlefish: Low-rank model training without all the tuning. Proceedings of Machine Learning and Systems, 5:578–605.

Yifan Wang, Zixuan Xu, Yiren Shen, Yujun Zhu, and Hui Xiong. 2024. Inscl: A data-efficient continual learning paradigm for fine-tuning large language models with instructions. arXiv preprint arXiv:2403.11435.

Yeming Wen and Swarat Chaudhuri. 2024. Batched low-rank adaptation of foundation models. In The Twelfth International Conference on Learning Representations.

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. arXiv preprint arXiv:2402.11417.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. arXiv preprint arXiv:2106.10199.

Jingfan Zhang, Yi Zhao, Dan Chen, Xing Tian, Huanran Zheng, and Wei Zhu. 2024. Milora: Efficient mixture of low-rank adaptation for large language models fine-tuning. In Findings of the Association for Computational Linguistics: EMNLP 2024, pages 17071–17084. Association for Computational Linguistics.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. 2023a. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning. Preprint, arXiv:2308.03303.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023b. Adaptive budget allocation for parameter-efficient fine-tuning. In The Eleventh International Conference on Learning Representations.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. arXiv preprint arXiv:1810.12885.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In Advances in Neural Information Processing Systems, volume 36, pages 46595–46623. Curran Associates, Inc.

Yitao Zhu, Zhenrong Shen, Zihao Zhao, Sheng Wang, Xin Wang, Xiangyu Zhao, Dinggang Shen, and Qian Wang. 2024. Melo: Low-rank adaptation is better than fine-tuning for medical image diagnosis. In 2024 IEEE International Symposium on Biomedical Imaging (ISBI), pages 1–5. IEEE.

A Tensor Algebra

To facilitate our analysis, we briefly present some tensor algebra preliminaries and refer the reader to (Sidiropoulos et al., 2017; Kolda and Bader, 2009) for further details.

A N-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is an N-way array indexed by $i_1, i_2, \ldots, i_N$ with elements $\mathcal{X}(i_1, i_2, \ldots, i_N)$. It consists of N types of modes: $\mathcal{X}(:, i_2, \ldots, i_N), \mathcal{X}(i_1, :, \ldots, i_N), \ldots, \mathcal{X}(i_1, i_2, \ldots, :)$.

A rank-one tensor $\mathcal{Z} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is the outer product of N vectors defined as:

$$\mathcal{Z} = \boldsymbol{a}_1 \circ \boldsymbol{a}_2 \circ \cdots \circ \boldsymbol{a}_N, \tag{11}$$

where $\boldsymbol{a}_1 \in \mathbb{R}^{I_1}$, $\boldsymbol{a}_2 \in \mathbb{R}^{I_2}, \ldots,$ $\boldsymbol{a}_N \in \mathbb{R}^{I_N}$ and $\circ$ denotes the outer product. The elementwise formula of the above expression is:

$$\mathcal{Z}(i_1, i_2, \ldots, i_N) = \boldsymbol{a}_1(i_1)\boldsymbol{a}_2(i_2) \cdots \boldsymbol{a}_N(i_N), \text{ for all} i_1, i_2, \ldots, i_N, \tag{12}$$

Any tensor can be realized as a sum of N-way outer products (rank one tensors), i.e.

$$\mathcal{X} = \sum_{r=1}^{R} \boldsymbol{a}_1^f \circ \boldsymbol{a}_2^f \circ \cdots \circ \boldsymbol{a}_N^f. \tag{13}$$

The above expression represents the *canonical polyadic decomposition* (CPD) or *parallel factor analysis* (PARAFAC) (Harshman and Lundy, 1994) of a tensor. The CPD elementwise representation is:

$$\mathcal{X}(i, j, k) = \sum_{r=1}^{R} \boldsymbol{A}_1(i_1, f)\boldsymbol{A}_2(i_2, f) \cdots \boldsymbol{A}_N(i_N, f), \tag{14}$$

where $\boldsymbol{A_n} = [\boldsymbol{a}_n^1, \boldsymbol{a}_n^2, \ldots, \boldsymbol{a}_n^F] \in \mathbb{R}^{I_n \times F}$, $n = 1, \ldots, N$ are called the low rank factors of the tensor. A tensor can be fully characterized by its latent factors, so we can represent a tensor by its CPD model as:

$$\mathcal{X} = [\![\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_N]\!]. \tag{15}$$

A tensor can be also represented as a set of matrices, by fixing all the modes but two as:

$$\mathcal{X}[:, :, i_3, \ldots, i_N] = \boldsymbol{A}_1 \left(\text{Diag}\left(\boldsymbol{A}_3(i_3, :)\right) \odot \cdots \odot \text{Diag}\left(\boldsymbol{A}_N(i_N, :)\right)\right) \boldsymbol{A}_2^T, \tag{16}$$

where $\text{Diag}\left(\boldsymbol{A}_n(i_n, :)\right)$ is the diagonal matrix with diagonal equal to $\boldsymbol{A}_N(i_n, :)$.

B Additional related work

Efficient Architectures Another relevant direction in reducing resource usage is using more efficient model architectures. Mixture of Experts (MoE) technique, implemented in models like Switch Transformers (Fedus et al., 2022) and GLaM (Du et al., 2022), has shown promise in scaling model capacity while maintaining computational efficiency by activating only relevant submodels for given inputs. Recent works (Buehler and Buehler, 2024; Zhang et al., 2024) have explored parameterizing experts, which often amount to different feed forward module parameters within the transformer block, using low rank adapters. modules(Bershatsky et al., 2024) have proposed that experts in Mixture of Experts (MoE) models could be also modeled jointly as a fourth order tensor $d\tilde{\mathcal{W}}_m \in \mathbb{R}^{d \times d \times L \times E}$, where E is the number of experts, but no tensor based models were explored in practice. There is also relevant work on non-transformer architectures, such as RWKV (Peng et al., 2023) and Mamba (Gu and Dao, 2023). PEFT methods for these architectures have also been explored (Kim et al., 2025; Ham et al., 2024).

Model Compression While these techniques differ from PEFT in that they focus on reducing the requirements of a trained model rather than efficient adaptation, they offer valuable insights for developing more efficient PEFT approaches. Pruning and quantization are key techniques for compressing neural networks, that have also been extensively applied to LLMs. Pruning removes less important weights, with some methods achieving high compression rates, e.g. (Ma et al., 2023). Quantization reduces weight precision, decreasing model size and also allowing more efficient operations (Lin et al., 2024a). The combination application of PEFT methods with quantization or pruning techniques to further improve efficiency has been explored, for example in (Dettmers et al., 2024; Benedek and Wolf, 2024).

Data efficient fine tuning. An alternative approach to reducing fine-tuning costs is to reduce the amount of data. In this direction, Few-shot and continual learning approaches have been shown to be effective in LLM fine-tuning tasks (Lin et al., 2024b; Wang et al., 2024).

Low Rank Training. Exploiting low rank structure to improve efficiency during both training and inference in deep models has long been studied (Sainath et al., 2013), and also combined with sparsity (Sprechmann et al., 2015). Recent advancements include Cutlefish (Wang et al., 2023) and ELRT (Sui et al., 2024).

C Other Tensor Low Rank Models in PEFT

C.1 Tensorizing individual weight updates.

(Yang et al., 2024) propose to each low rank matrix in a LoRA adapter. Explicitly, if for a single weight update $d\tilde{\boldsymbol{W}} \in \mathbb{R}^{d \times d}$, is first expressed using the low rank matrix model

$$d\tilde{\boldsymbol{W}} = \frac{\alpha}{r}\boldsymbol{A}\boldsymbol{B}^T, \quad \boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{d \times r}, \tag{17}$$

Then, both low rank matrix factors $\boldsymbol{A}$ and $\boldsymbol{B}$ are expressed as order-D tensors $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{k_1 \times \ldots \times k_D}$, where the chosen dimensions must satisfy $\prod_{i=1}^{D} k_i = d^2$. The condition $k_i \geq r$ for all i also imposes a limit on the order of this tensor: $n \leq \log_r(d)$. Since $n > 2$ is required for LoreTTA to be potentially more efficient than

LoRA, $r > \sqrt{d}$, which This can be limiting in practice since $n > 2$ is required for LoReTTA to be potentially more efficient than LoRA. This implies that $r > \sqrt{d}$, which can be limiting in practice. For example, $r \leq 64$ for a Llama-7B model.

These tensorized updates are then parameterized using a low rank Tensor-Train model with equal ranks across all factors. Explicitly:

$$d\tilde{\mathcal{A}} = \prod_{i=1}^{D} \mathcal{G}_i, \quad \mathcal{G}_i \in \mathcal{R}^{r \times k_i \times r}.$$

C.2 Parameter sharing across layers

Both (Jie and Deng, 2023) in the context of vision transformers and (Bershatsky et al., 2024) for LLMs have proposed to represent the updates of each fine-tuned attention matrix $(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \boldsymbol{P})$ across all layers as a tensor $d\tilde{\mathcal{W}}_m \in \mathbb{R}^{d \times d \times L}$. (Bershatsky et al., 2024) parametrize it using a Tucker-2 model

$$d\tilde{\mathcal{W}}_m = \boldsymbol{G} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B}, \tag{18}$$

where $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{d \times r}$ and $\boldsymbol{G} \in \mathbb{R}^{L \times r \times r}$.

D Parameter efficiency gains breakdown.

We provide a breakdown of the parameter savings achieved by our proposed method, LoRTA, compared to LoRA, by parameterizing the weight updates using low-rank tensor decompositions at different granularities – i.e., shared modes–. Table 8 summarizes the dimensions of the update tensors, the number of update tensors used, and the corresponding parameter savings when the tensor rank r matches the tensor rank of LoRA rank r. The first row corresponds to LoRA.

To fairly compare the parameter efficiency of LoRTA with LoRA, we adjust the tensor rank in LoRTA to match the effective total tensor rank in LoRA, which is $r' = r \times 4L$ due to LoRA applying a rank r update to each of the $4L$ matrices individually. For a given tensor rank, LoRTA reduces the number of parameters from scaling $8dLr$ in LoRA to $4L(d(1+1/h)+h+L+4)r$ in LoRTA (usually $d \gg L$ and $d \gg h$), achieving substantial parameter savings without compromising expressive power. For example, this amounts to a 47.6% reduction in a LLaMA2 7B model.

E Experimental details

In this appendix, we provide further details on the experiments presented in the main paper.

E.1 NLU

In our GLUE experiments we implemented our method using Huggingface's PEFT, VeRA (Kopiczko et al., 2023) and LoreTTA (Yang et al., 2024) codebases. Hyperparameters for each of the three settings reported are detailed below.

E.2 Instruction tuning

For instruction tuning experiments we utilized Lightning AI's LitGPT codebase and training recipe. Hyperparameters are detailed below.

E.3 DPO

For preference optimization experiments we utilized using Huggingface trl library's dpo implementation and example script. Hyperparameters are detailed below.

E.4 Protein Folding

For protein folding experiments, we utilized OpenFold (Ahdritz et al., 2024) training code and datasets. The following modifications were made to the ESMFold model architecture due to limited compute resources: a) utilize 12 Evoformer layers instead of the 48 used in (Lin et al., 2023) b) utilize ESM-2 35M instead of ESM-2 3B c) maintain outer product mean implementation from (Jumper et al., 2021). Optimizer and learning rate scheduler were identical to (Jumper et al., 2021). Models were trained for 850,000 steps with batch size of 32. Validation metrics were computed using the validation set from (Ahdritz et al., 2024).

Preliminary experiments revealed that higher values of α yield better results in this setting. α for LoRA and LoRTA experiments was then selected in multiple stages. Initially, models were trained with α values of $256 \times r$ and $128 \times r$, and the best-performing model was chosen. If both configurations diverged, α was halved, and models were retrained with the next lower pair (e.g., $64 \times r$ and $32 \times r$). This halving process continued until a convergent model was found. See Table 14 for the selected α values across experiments.

F Additional results

F.1 Instruction Tuning

To further evaluate the fine-tuned models, we use MT-Bench (Zheng et al., 2023), an LLM-as-a-judge benchmark. MT-Bench assesses multi-turn conversational and instruction-following abilities on 80 open-ended questions, covering diverse capabilities such as roleplaying, reasoning, coding and information retrieval. GPT-4 is used to score the outputs of the model on a scale of one to ten.

As shown in Figure 3, LoRTA can almost match average performance despite using just 1/5th of the parameters (r=48). Unlike the loss observed in the Alpaca dataset, performance does not increase monotonically, potentially due to overfitting. Moreover, performance varies across tasks. For example, most LoRTA models surpass LoRA in reasoning but fall short in writing.

F.2 Preference Optimization

As shown in Figure 4 LoRTA exhibited non-monotonic performance across ranks. This suggests that further hyperparameter tuning may be necessary to stabilize its

Table 8: Breakdown of the parameter savings against LoRA by mode, i.e., parameter sharing dimension. This is the number of parameters required to represent an update with thensor rank r.

Added Modes	Update Tensor Dimensions	Number of Update Tensors	Parameter Savings
	$d \times d$	$4L$	0
Heads	$d \times \frac{d}{H} \times H$	$4L$	$1 - \frac{d\left(1+\frac{1}{H}\right)+H}{2dr}$
Heads, QKVP	$d \times \frac{d}{H} \times H \times 4$	L	$1 - \frac{d\left(1+\frac{1}{H}\right)+H+4}{2dr}$
Heads, QKVP, Layers	$d \times \frac{d}{H} \times H \times 4 \times L$	1	$1 - \frac{d\left(1+\frac{1}{H}\right)+H+4+L}{2dr}$

Hyperparameter	Value
α	16
Learning Rate	[2E-3, 5E-4]
Scheduler	Constant
Optimizer	AdamW
Number of Epochs	20
Batch Size	[16, 32]
Warmup Steps	500

Table 9: Hyperparameter configurations for RoBERTa Base on the GLUE benchmark following the setup reported by (Yang et al., 2024), where only the batch size and learning rate are tuned for each task, selecting between two values based on validation performance. All other hyperparameters match those reported by (Yang et al., 2024).

Hyperparameter	Value
α	[0.5 1.0 2.0 8.0]
Learning Rate	[5e-4, 1e-3, 5e-3, 1e-2]
Scheduler	Linear
Optimizer	AdamW
Number of Epochs	20
Batch Size	32
Warmup Ratio	0.06

Table 10: Hyperparameter configurations for RoBERTa Base on the GLUE benchmark following (Bershatsky et al., 2024). A grid-search to set the learning rate and scale parameter for each task is conducted across the specified values.

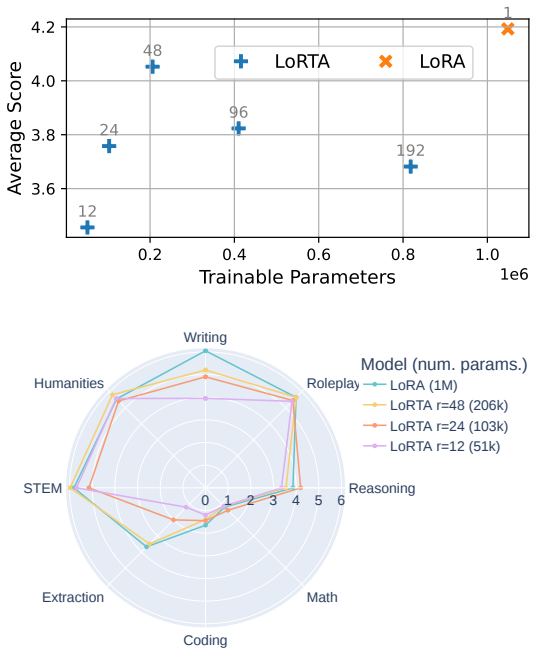


Figure 3. Performance on MT-Bench (Zheng et al., 2023) for Llama2-7b (Touvron et al., 2023) models fine-tuned with LoRA and LoRTA. Higher is better. **Left**: Average score across all questions vs number of trainable parameters. Numbers on top of markers denote the adapter rank. **Right**: Average score per task.

performance. Although we did not tune hyperparameters, most ranks still outperformed LoRA with significantly fewer parameters.

We further evaluated the fine-tuned models on the LLM-as-a-judge MT-benchmark. In this setting, LoRTA consistently outperformed LoRA across all ranks, including at rank 2 where it had shown higher DPO loss on the preference dataset. This improvement suggests enhanced out-of-distribution generalization capabilities for LoRTA adapters since MT-bench differs from the training dataset.

Figure 6 shows that Validation gains were primarily driven by reduced training error, though generalization slightly worsened, particularly at rank 2. On the other hand, as already mentioned, MT-bench performance was comparable o superior for LoRTA across all ranks, as shown in Figure 5.

F.3 Protein Folding

In figure 7 we include higher ranks for LoRTA in the protein folding experiment. However, note that increasing the rank beyond 1 and even matching the number of parameters in LoRA does not result in performance improvements. We also include train error to show that although LoRA rank 1 shows a performance improvement (Mean LDDT-Cα) in the training set, it shows a larger generalization gap.

G Training time and memory

Hyperparameter	SST-2	MRPC	CoLA	QNLI	RTE	STS-B
Optimizer	AdamW					
Warmup Ratio	0.06					
LR Schedule	Linear					
Epochs	10	40	40	20	40	20
Learning Rate (Head)	6E-3	3E-3	6E-3	2E-4	2E-3	2E-3
Learning Rate (Encoder)	1E-2	1E-2	1E-2	1E-2	2E-2	2E-2
Batch Size	32					

Table 11: Hyperparameter configurations for RoBERTa large on the GLUE benchmark. All other hyperparameters are taken from (Kopiczko et al., 2023).

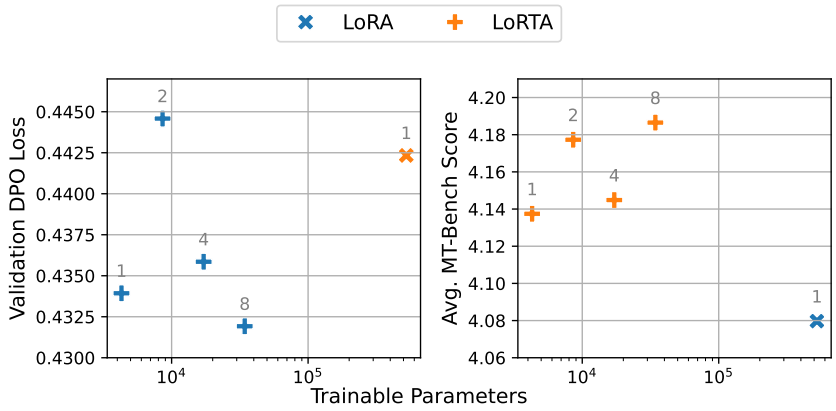


Figure 4. (Left) Mean DPO loss on held-out data from the orca dpo pairs dataset vs number of trainable parameters, lower is better. (Right) MT-Bench average scores Scores vs number of trainable parameters, higher is better.

Parameter	**Value**
α	16
Learning Rate	0.01
Scheduler	Cosine
Optimizer	AdamW
Weight Decay	0.01
Number of Epochs	1
Steps	51000
Batch Size	16
Warmup Steps	318

Table 12: Hyperparameter configurations for LLama2-7B on the Alpaca dataset.

Table 13: Hyperparameter configurations for LLama2-7B on intel orca DPO pairs.

Parameter	**Value**
α	16
Learning Rate	0.00005
Scheduler	Cosine
Optimizer	AdamW
Weight Decay	0
Number of Epochs	1
Batch Size	16
Warmup Steps	200

Table 14: Selected α and LDDT-CA for protein folding models.

Model	α	**Validation LDDT-Cα**
LoRA (r = 1)	128	0.668
LoRTA (r = 64)	128	0.663
LoRTA (r = 8)	256	0.667
LoRTA (r = 1)	2	0.656

During training, the reduction in GPU memory usage from shrinking optimizer states is marginal for parameter reductions beyond LoRA. Memory consumption in these cases is dominated by activations and caches stored during forward and backpropagation. Additional memory savings could be achieved by compressing activations or gradients, leveraging the low-rank structure of updates, or dynamically recomputing them. While our model features fewer trainable parameters and could theoretically benefit from the efficient tensor CP structure, such as faster training and lower memory usage, these advantages are not yet realized due to the limitations of our current implementation. We leave these optimizations for future work. However, the reduced parameter count already provides lower storage requirements and faster I/O.

We conducted hardware profiling to compare the per-

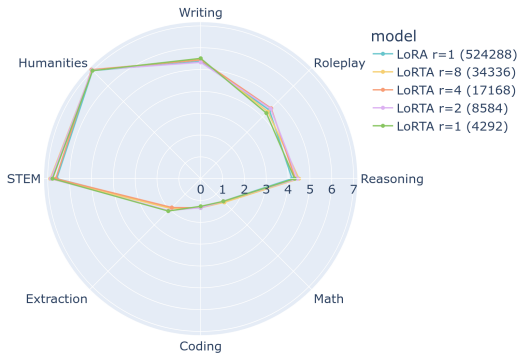


Figure 5. Performance on MT-Bench (Zheng et al., 2023) for llama2-7b (Touvron et al., 2023) models fine-tuned with LoRA and LoRTA using dpo on intel orca pairs. Average score per task. Higher is better.

formance of our LoRTA implementation against LoRA using HuggingFace PEFT. The results demonstrate negligible differences in resource consumption between the two methods. The slight gap in training time for LoRTA can be addressed through further optimizations, ranging from leveraging tools like Torch Compile, to implementing our CP tensor adapter model more efficiently.

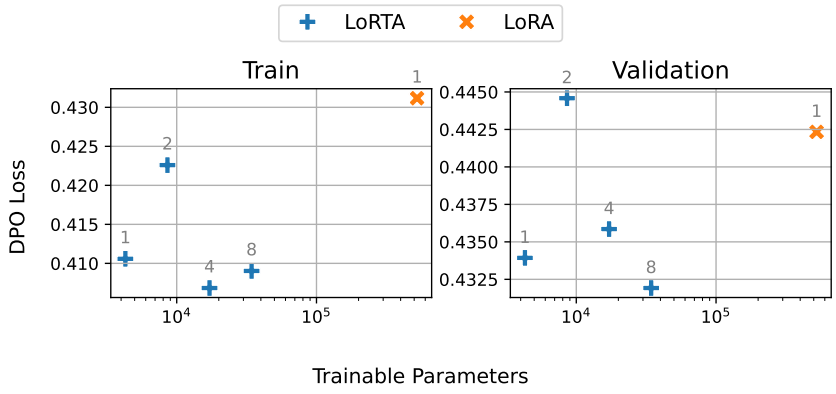


Figure 6. Mean DPO loss on the training (Left) and on held-out data (Right) from the orca dpo pairs dataset vs number of trainable parameters, lower is better.

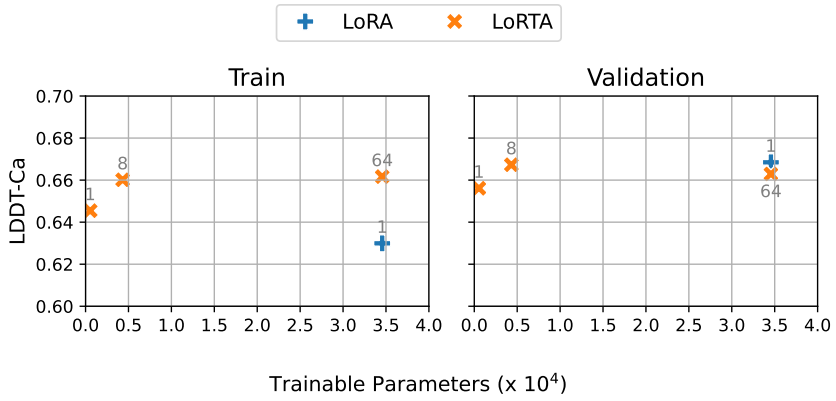


Figure 7. Mean LDDT-Cα on held-out (left) and train sets. Higher is better. LoRTA rank 1 is competitive with LoRA rank 1 on the test set despite having 64x fewer parameters. Numbers on top of markers denote the adapter rank.

Rank	Method	GPU Mem. (GB)	FLOPs (avg)	MACs (avg)	Time (s/step)
4	LoRA	12.84	272	136	0.07
	LoRTA	12.88	272	136	0.14
64	LoRA	13.08	276	138	0.09
	LoRTA	12.98	273	136	0.14

Table 15: Maximum GPU memory usage (GB), average FLOPs(GB), MACs(GB), and training time (seconds per step) for LoRA and LoRTA.