A TENSOR ALGEBRA

To facilitate our analysis, we briefly present some tensor algebra preliminaries and refer the reader to Sidiropoulos et al. (2017); Kolda & 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:

$$\mathcal{Z} = \boldsymbol{a}_1 \circ \boldsymbol{a}_2 \circ \cdots \circ \boldsymbol{a}_N, \tag{15}$$

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:

$$\mathcal{Z}(i_1, i_2, \ldots, i_N) = \boldsymbol{a}_1(i_1)\boldsymbol{a}_2(i_2) \cdots \boldsymbol{a}_N(i_N), \text{ for all} i_1, i_2, \ldots, i_N, \tag{16}$$

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

$$\mathcal{X} = \sum_{r=1}^{R} \boldsymbol{a}_1^f \circ \boldsymbol{a}_2^f \circ \cdots \circ \boldsymbol{a}_N^f. \tag{17}$$

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

$$\mathcal{X}(i, j, k) = \sum_{r=1}^{R} \boldsymbol{A}_1(i_1, f)\boldsymbol{A}_2(i_2, f) \cdots \boldsymbol{A}_N(i_N, f), \tag{18}$$

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:

$$\mathcal{X} = [\![\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_N]\!] \,. \tag{19}$$

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

$$\begin{aligned} &\mathcal{X}[:, :, i_3, \ldots, i_N] = \\ &\boldsymbol{A}_1 \left(\text{Diag}\left(\boldsymbol{A}_3\left(i_3, :\right)\right) \odot \cdots \odot \text{Diag}\left(\boldsymbol{A}_N\left(i_N, :\right)\right)\right) \boldsymbol{A}_2^T, \end{aligned} \tag{20}$$

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

B ADDITIONAL RELATED WORK

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). Knowledge distillation is an alternative approach that involves transferring knowledge from a large "teacher" model to a smaller "student" model (Gu 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 Cuttlefish (Wang et al., 2023) and ELRT (Sui et al., 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).

Efficient Architectures Another relevant direction in 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 sub-models for given inputs. There is also relevant work on non-transformer architectures, such as RWKV (Peng et al., 2023) and Mamba (Gu & Dao, 2023), which combines the strengths of RNNs and Transformers to achieve efficient inference and training.

C 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. The table below 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.

Table 2: Update Tensor Modes, Parameters, and Savings

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	$1 - \frac{d\left(1+\frac{1}{H}\right)+H+4+L}{2dr}$

D EXPERIMENTAL DETAILS

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

D.1 NLU

In our GLUE experiments we implemented our method using Huggingface's PEFT and VeRA's codebase, the hyperparameters are detailed below.

Model	Hyperparameter	SST-2	MRPC	CoLA	QNLI	RTE	STS-B
	Optimizer	AdamW					
	Warmup Ratio	0.06					
	LR Schedule	Linear					
	Init. of Shared Matrices	Kaiming Uniform					
	Initial Value of d	0.1					
	LoRTA Rank	4					
	Epochs	60	30	80	25	160	80
	Learning Rate (Head)	4E-3	4E-3	1E-2	4E-3	1E-2	1E-2
	Learning Rate (LoRTA)	4E-3	1E-2	1E-2	1E-2	4E-3	5E-2
	Max Seq. Len.	512					
	Batch Size	64					
	LoRTA Rank	8					
	Epochs	10	40	40	20	40	20
	Learning Rate (Head)	6E-3	3E-3	6E-3	2E-4	2E-3	2E-3
	Learning Rate (LoRTA)	1E-2	1E-2	1E-2	1E-2	2E-2	2E-2
	Max Seq. Len.	128					
	Batch Size	32					

Table 3: Hyperparameter configurations for different model sizes on GLUE benchmark. *Optimizer*, *Warmup Ratio*, and *LR Schedule* are taken from Hu et al. (2021), all other hyperparameters except for learning rate are taken from Kopiczko et al. (2023).

D.2 INSTRUCTION TUNING

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

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

D.3 DPO

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

Table 5: 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

D.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 6 for the selected α values across experiments.

Table 6: 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

E ADDITIONAL RESULTS

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 7.

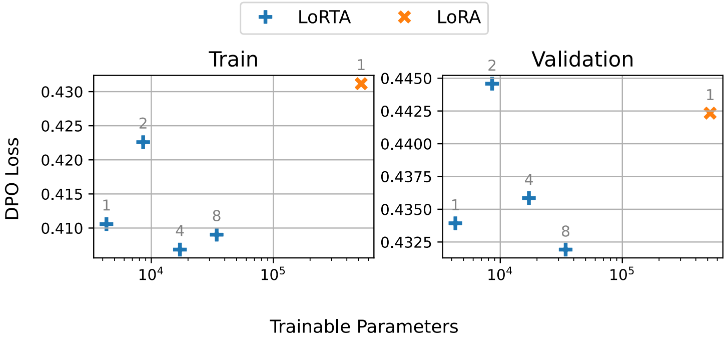

Figure 6. Mean DPO loss on the training (Left) and on held-out data (Right) from the orca dpo pairs dataset vs number of trainable parameters, lower is better.

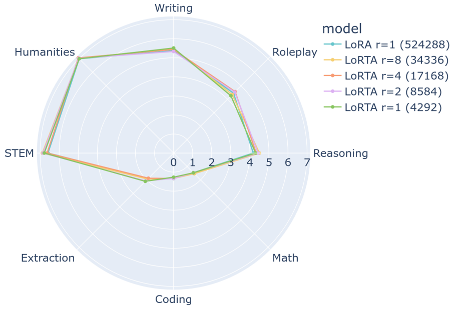

Figure 7. Performance on MT-Bench Zheng et al. (2023) for llama2-7b Touvron et al. (2023) models fine-tuned with LoRA and LoRTA using dpo on intel orca pairs. Average score per task. Higher is better.

F NOTATION

Our notation is summarized in Table 5.

Table 5: Overview of notation.

a	$\triangleq$	scalar
$\boldsymbol{a}$	$\triangleq$	vector
$\boldsymbol{A}$	$\triangleq$	matrix
$\boldsymbol{A}^T$	$\triangleq$	transpose of matrix $\boldsymbol{A}$
$\boldsymbol{A}_k$	$\triangleq$	$\boldsymbol{A}[k,:]^T$, k-th row of matrix $\boldsymbol{A}$
$\boldsymbol{a}_k$	$\triangleq$	$\boldsymbol{A}[:,k]$, k-th column of matrix $\boldsymbol{A}$
$\boldsymbol{U}$	$\triangleq$	eigenvector matrix
$\boldsymbol{U}[k,:]$	$\triangleq$	k-th row of $\boldsymbol{U}$ (row vector)
$\boldsymbol{U}[:,k]$	$\triangleq$	k-th column of $\boldsymbol{U}$