

LORA-CRAFT: Cross-layer Rank Adaptation via Frozen Tucker Decomposition of Pre-trained Attention Weights*

Kasun Dewage, Marianna Pensky, Shankhadeep Mondal & Suranadi De Silva
University of Central Florida
Orlando, FL 32816, USA
{KasunTharuka.Dewage, Marianna.Pensky}@ucf.edu
{shankhadeep.mondal, su966204}@ucf.edu

Abstract

We introduce LoRA-CRAFT (Cross-layer Rank Adaptation via Frozen Tucker), abbreviated CRAFT throughout, an extremely parameter-efficient fine-tuning (PEFT) method that applies Tucker tensor decomposition to pre-trained attention weight matrices stacked across transformer layers and trains only small square adaptation matrices on the resulting frozen Tucker factors. Existing tensor-based PEFT methods decompose *gradient updates*: LoTR applies Tucker decomposition with shared factor matrices, while SuperLoRA groups and reshapes ΔW across layers before applying Tucker decomposition. Separately, methods such as PiSSA apply SVD to *pre-trained weights* but operate independently per layer. CRAFT bridges these two lines of work: it performs full Tucker decomposition via Higher-Order SVD (HOSVD) directly on *pre-trained weights* organized as cross-layer 3D tensors, freezes all resulting factors, and adapts the model through lightweight trainable transformations applied to each factor matrix. Experiments on the GLUE benchmark using RoBERTa-base and RoBERTa-large, as well as commonsense reasoning benchmarks using LLaMA2-7B and LLaMA3-8B, demonstrate that CRAFT achieves competitive performance with existing methods while requiring only **extremely low Tucker adaptation parameters**. On LLaMA3-8B, CRAFT exceeds the average accuracy of LoRA **using hundreds of times fewer parameters**; on LLaMA2-7B the same holds at a 0.252M budget. Our results suggest that CRAFT’s efficiency advantage grows with model scale, as the frozen Tucker factors better capture the richer cross-layer structure of larger pre-trained models.

1 Introduction

The transformer architecture (Vaswani et al., 2017) has become the dominant paradigm in natural language processing, but adapting large pre-trained models to downstream tasks remains computationally prohibitive. Parameter-efficient fine-tuning (PEFT) methods address this challenge by training only a small subset of parameters while keeping the bulk of the model frozen. LoRA (Hu et al., 2022) pioneered the approach of decomposing weight updates into low-rank matrices, reducing trainable parameters from millions to thousands. However, LoRA and its variants treat each weight matrix independently, missing a crucial insight: attention mechanisms in transformers exhibit strong *multi-way correlations* across layers. Recently, two complementary research directions have begun to exploit this observation.

The first line of work analyzes *gradient updates* using tensor decomposition techniques. LoTR (Bershtsky et al., 2024) represents gradient updates as a tensor and applies Tucker decomposition with shared factor matrices across layers. SuperLoRA (Chen et al., 2024) extends this by grouping and reshaping ΔW across multiple layers before applying Tucker decomposition. VeRA (Kopiczko et al., 2024) shares random frozen matrices across lay-

*Code and logs will be available: https://github.com/Kasun-Dewage/Lora-Craft_2026

ers, and LoRETTA (Yang et al., 2024) employs tensor-train decomposition for ultra-low-parameter fine-tuning. The second approach decomposes the *pre-trained weights themselves*. PiSSA (Meng et al., 2024) applies SVD to pre-trained weight matrices, initializing LoRA adapters with the principal singular values and vectors while freezing the residual. However, PiSSA operates independently on each layer’s weight matrix, missing cross-layer correlations.

Before proceeding, we briefly clarify Tucker decomposition terminology. The *Tucker decomposition* of an N th-order tensor expresses it as a core tensor multiplied by a factor matrix along each mode. A *Tucker-2* decomposition retains factor matrices for only two of three modes; for instance, LoTR (Bershtsky et al., 2024) employs Tucker-2. CRAFT employs Tucker-3 (full Tucker decomposition for third-order tensors), compressing all three modes simultaneously.

We propose CRAFT, which unifies these two directions by performing tensor decomposition on pre-trained weights organized as cross-layer tensors. CRAFT differs from all prior approaches in two key ways:

1. Cross-layer decomposition of pre-trained weights. Like PiSSA, CRAFT decomposes pre-trained weights rather than gradient updates. But unlike PiSSA’s per-layer matrix SVD, CRAFT stacks weights across layers into 3D tensors and applies full Tucker-3 decomposition, simultaneously capturing cross-layer, output-dimension, and input-dimension patterns.

2. Frozen factors with trainable adaptation matrices. While SuperLoRA and LoTR train the Tucker factors or core tensors themselves, CRAFT freezes *all* Tucker factors (including the core tensor) obtained from HOSVD and introduces small square adaptation matrices $J^{(1)}, J^{(2)}, J^{(3)}$ as the *only* trainable parameters, initialized near identity. This yields a trainable parameter count independent of both model dimension d and number of layers N_L , for fixed Tucker ranks.

Following the standard LoRA convention (Hu et al., 2022), CRAFT adapts Q and V projections. The choice of Q over K is motivated by gauge invariance of the bilinear form $QK^\top$ and empirical PCA analysis; details are provided in Appendix A.1. This is our main configuration; we additionally report a variant, CRAFT*, in which all four projections $\{Q, K, V, O\}$ are adapted.

Advantages of CRAFT. The CRAFT framework offers several distinct advantages: **(a)** With a **small number of Tucker adaptation parameters**, which at fixed ranks is **independent** of both the model dimension d and the number of layers N_L , it achieves accuracy comparable to methods with significantly larger parameter budgets. **(b)** Because only $n_p(r_1^2 + r_2^2 + r_3^2)$ parameters carry gradients, the **gradient and optimizer-state memory** is negligible relative to LoRA or full fine-tuning. We do not claim a wall-clock speed-up. Under identical LLaMA3-8B profiling conditions, CRAFT’s smallest configuration (0.018M trainable parameters) was approximately 0.5% faster than LoRA (27.3M), the main (32, 128, 128) configuration (0.067M) was 1.4% slower within the observed 3–4% run-to-run variation and the largest configuration (0.252M) was 6.8% slower, whereas DoRA (27.3M) required approximately 71% more time per optimizer step than LoRA.

(c) The method is **theoretically grounded** with approximation guarantees inherited from HOSVD. Specifically, while in the case of additive LoRA adaptations, choosing the ranks of adaptations is a separate and rather cumbersome task, CRAFT preserves the effective ranks of the fully known matrices W . Therefore, one can provide theoretically sound HOSVD ranks (see Theorem 1 in Section 4). A more detailed comparison with related work is presented in Section 3. A design comparison of tensor-based PEFT methods is provided in Table 3 (Appendix A), along with a visual taxonomy (Figure 2) and parameter scaling analysis (Figure 3).

2 The CRAFT methodology

In what follows, we denote tensors by calligraphic letters. We denote elements of matrix A or tensor $\mathcal{W}$ by $A(i, j)$ and $\mathcal{W}(i, j, k)$. Following (Kolda & Bader, 2009), we introduce the

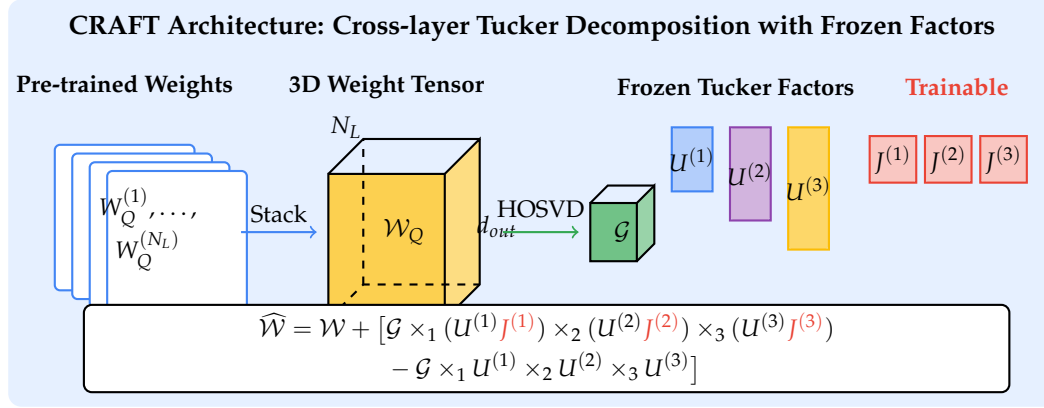


Figure 1: CRAFT architecture overview. Pre-trained attention weights (Q, V) are stacked across N_L layers into 3D tensors. HOSVD decomposes each tensor into a core tensor $\mathcal{G}$ and factor matrices $U^{(1)}, U^{(2)}, U^{(3)}$. All decomposition factors are **frozen**. Adaptation occurs only through small square matrices $J^{(1)} \in \mathbb{R}^{r_1 \times r_1}$, $J^{(2)} \in \mathbb{R}^{r_2 \times r_2}$, $J^{(3)} \in \mathbb{R}^{r_3 \times r_3}$ (shown in red), initialized near identity. This yields a trainable parameter count of $n_p(r_1^2 + r_2^2 + r_3^2)$, where n_p is the number of adapted projection types ($n_p = 2$ for Q and V)—**independent of model dimension d and depth N_L at fixed Tucker ranks**.

following notation. A third-order tensor $\mathcal{W} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is a three-dimensional array with elements $w_{i_1 i_2 i_3}$.

Definition 1 (Mode- n Unfolding). The mode- n unfolding (matricization) of tensor $\mathcal{W}$, denoted $W_{(n)}$, arranges the mode- n fibers as columns of a matrix. For a third-order tensor: $W_{(1)} \in \mathbb{R}^{I_1 \times I_2 I_3}$, $W_{(2)} \in \mathbb{R}^{I_2 \times I_1 I_3}$, $W_{(3)} \in \mathbb{R}^{I_3 \times I_1 I_2}$.

Definition 2 (Mode- n Product). The mode- n product of tensor $\mathcal{W} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ with matrix $U \in \mathbb{R}^{J \times I_n}$, denoted $\mathcal{W} \times_n U$, yields a tensor of size $I_1 \times \dots \times I_{n-1} \times J \times I_{n+1} \times \dots \times I_N$.

In particular, for $n = 2$, one has $(\mathcal{W} \times_2 U)(i_1, j, i_3) = \sum_{i_2=1}^{I_2} \mathcal{W}(i_1, i_2, i_3) U(j, i_2)$.

Definition 3 (Tucker Decomposition). The Tucker decomposition of $\mathcal{W} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ with multilinear rank (r_1, r_2, r_3) is: $\mathcal{W} \approx \mathcal{G} \times_1 U^{(1)} \times_2 U^{(2)} \times_3 U^{(3)}$ where $\mathcal{G} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ is the core tensor and $U^{(n)} \in \mathbb{R}^{I_n \times r_n}$, $n = 1, 2, 3$, are factor matrices with orthonormal columns.

Given a transformer with N_L layers, CRAFT proceeds in three stages: tensor construction, frozen HOSVD decomposition, and adaptation via trainable matrices. The complete procedure is described in Algorithm 1.

Stage 1: Tensor Construction. For each projection type $\alpha \in \mathcal{A}$, we construct a 3D tensor by stacking the pre-trained attention weight matrices across layers:

$$\mathcal{W}_\alpha = \text{stack}(W_\alpha^{(1)}, W_\alpha^{(2)}, \dots, W_\alpha^{(N_L)}) \in \mathbb{R}^{N_L \times d_{out} \times d_{in}}$$

This construction treats the collection of weight matrices as a single multi-way array, enabling decomposition methods that capture correlations simultaneously across layers (mode-1), output dimensions (mode-2), and input dimensions (mode-3).

Stage 2: HOSVD Decomposition, Residual Computation, and Freezing. We compute the Tucker-3 decomposition of each $\mathcal{W}_\alpha$ via HOSVD (De Lathauwer et al., 2000). For each mode $n \in \{1, 2, 3\}$, we compute the mode- n unfolding $(\mathcal{W}_\alpha)_{(n)}$, take its truncated SVD to obtain the first r_n left singular vectors $U_\alpha^{(n)}$, and then compute the core tensor $\mathcal{G}_\alpha = \mathcal{W}_\alpha \times_1 (U_\alpha^{(1)})^\top \times_2 (U_\alpha^{(2)})^\top \times_3 (U_\alpha^{(3)})^\top$. We use standard truncated HOSVD computed from the three mode unfoldings, without a subsequent HOOI refinement step, so that

Algorithm 1 CRAFT: Cross-layer Rank Adaptation via Frozen Tucker

Require: Set of adapted projections $\mathcal{A} \subseteq \{Q, K, V, O\}$ with $n_p = |\mathcal{A}|$ (default $\mathcal{A} = \{Q, V\}$); pre-trained weights $\{W_\alpha^{(\ell)}\}_{\ell=1}^{N_L}$ for $\alpha \in \mathcal{A}$, $W_\alpha^{(\ell)} \in \mathbb{R}^{d_{out} \times d_{in}}$; Tucker ranks (r_1, r_2, r_3) with $r_1 \leq N_L$; scaling $s > 0$; dropout rate $p \geq 0$ on $J^{(n)}$; learning rate η ; loss $\mathcal{L}$

Ensure: Adapted weight tensor $\widehat{\mathcal{W}}_\alpha$; frozen factors $U^{(n)}$, core $\mathcal{G}$; trained matrices $J^{(n)}$, $n = 1, 2, 3$

- 1: **// Stage 1: Tensor Construction**
- 2: **for** each projection $\alpha \in \mathcal{A}$ **do**
- 3: Stack: $\mathcal{W}_\alpha \leftarrow \text{stack}(W_\alpha^{(1)}, \dots, W_\alpha^{(N_L)}) \in \mathbb{R}^{N_L \times d_{out} \times d_{in}}$
- 4: **end for**
- 5: **// Stage 2: HOSVD Decomposition (one-time)**
- 6: **for** each $\alpha \in \mathcal{A}$ and $n = 1, 2, 3$ **do**
- 7: Compute mode- n unfolding $W_{(n)}$; set $U_\alpha^{(n)} \leftarrow$ first r_n left singular vectors of $W_{(n)}$
- 8: **end for**
- 9: **for** each $\alpha \in \mathcal{A}$ **do**
- 10: $\mathcal{G}_\alpha \leftarrow \mathcal{W}_\alpha \times_1 (U_\alpha^{(1)})^\top \times_2 (U_\alpha^{(2)})^\top \times_3 (U_\alpha^{(3)})^\top$; $\mathcal{R}_\alpha \leftarrow \mathcal{G}_\alpha \times_1 U_\alpha^{(1)} \times_2 U_\alpha^{(2)} \times_3 U_\alpha^{(3)}$
- 11: **Freeze:** $\mathcal{W}_\alpha, \mathcal{G}_\alpha, \mathcal{R}_\alpha, U_\alpha^{(1)}, U_\alpha^{(2)}, U_\alpha^{(3)}$ (in practice, store the single buffer $\mathcal{W}_\alpha - \mathcal{R}_\alpha$)
- 12: **end for**
- 13: **// Stage 3: Initialize Trainable Adaptation**
- 14: **for** each $\alpha \in \mathcal{A}$, each $n \in \{1, 2, 3\}$ **do**
- 15: $J_\alpha^{(n)} \leftarrow I_{r_n} + \epsilon \cdot E^{(n)}$, $E_{ij}^{(n)} \sim \text{i.i.d. } \mathcal{N}(0, \sigma^2)$
- 16: **end for**
- 17: **// Stage 4: Training (residual-preserving)**
- 18: **while** not converged **do**
- 19: **for** each $\alpha \in \mathcal{A}$ **do**
- 20: $\mathcal{T}_\alpha \leftarrow \mathcal{G}_\alpha \times_1 (U_\alpha^{(1)} \tilde{J}_\alpha^{(1)}) \times_2 (U_\alpha^{(2)} \tilde{J}_\alpha^{(2)}) \times_3 (U_\alpha^{(3)} \tilde{J}_\alpha^{(3)})$, $\tilde{J}_\alpha^{(n)} = \text{Dropout}_p(J_\alpha^{(n)})$
- 21: $\widehat{\mathcal{W}}_\alpha \leftarrow \mathcal{W}_\alpha + s(\mathcal{T}_\alpha - \mathcal{R}_\alpha)$; extract $\widehat{W}_\alpha^{(\ell)} \leftarrow \widehat{\mathcal{W}}_\alpha(\ell, :, :)$
- 22: **end for**
- 23: Forward pass with $\{\widehat{W}_\alpha^{(\ell)}\}_{\alpha \in \mathcal{A}}$ (projections outside $\mathcal{A}$ frozen); compute $\mathcal{L}$; update $J_\alpha^{(n)} \leftarrow J_\alpha^{(n)} - \eta \nabla_{J_\alpha^{(n)}} \mathcal{L}$
- 24: **end while**
- 25: **return** $\widehat{\mathcal{W}}_\alpha; \{U_\alpha^{(n)}, \mathcal{G}_\alpha, J_\alpha^{(n)}\}, n = 1, 2, 3$

Theorem 1 applies verbatim. Note that the mode-1 unfolding has only N_L rows, so the mode-1 rank is necessarily capped at $r_1 \leq N_L$. Since the Tucker decomposition with ranks $(r_1, r_2, r_3) < (N_L, d_{out}, d_{in})$ is not exact, the initial reconstruction

$$\mathcal{R}_\alpha = \mathcal{G}_\alpha \times_1 U_\alpha^{(1)} \times_2 U_\alpha^{(2)} \times_3 U_\alpha^{(3)} \quad (1)$$

is *not* equal to $\mathcal{W}_\alpha$ in general. We therefore retain the residual $\mathcal{W}_\alpha - \mathcal{R}_\alpha$; together with the adapted Tucker reconstruction, this permits exact recovery of the pre-trained weights when $J_\alpha^{(n)} = I_{r_n}$. With the reported near-identity initialization $J_\alpha^{(n)} = I_{r_n} + 10^{-3} E^{(n)}$, the model starts close to, but not exactly at, the pre-trained solution. The residual buffer $\mathcal{W}_\alpha - \mathcal{R}_\alpha$, the core $\mathcal{G}_\alpha$, and the factor matrices $U_\alpha^{(1)}, U_\alpha^{(2)}, U_\alpha^{(3)}$ are frozen.

Stage 3: Residual-Preserving Adaptation. We introduce trainable square matrices $J_\alpha^{(n)} \in \mathbb{R}^{r_n \times r_n}$, $n = 1, 2, 3$, initialized near identity:

$$J_\alpha^{(n)} = I_{r_n} + \epsilon \cdot E, \quad E_{ij} \sim \mathcal{N}(0, \sigma^2) \quad (2)$$

with $\epsilon = 10^{-3}$ and $\sigma = 1$; equivalently, every entry of $J_\alpha^{(n)}$ is perturbed by i.i.d. Gaussian noise of standard deviation 10^{-3} . The adapted weight tensor is computed via the *residual-preserving* formula:

$$\widehat{\mathcal{W}}_\alpha = \mathcal{W}_\alpha + s(\mathcal{T}_\alpha - \mathcal{R}_\alpha), \quad (3)$$

where

$$\mathcal{T}_\alpha = \mathcal{G}_\alpha \times_1 (U_\alpha^{(1)} J_\alpha^{(1)}) \times_2 (U_\alpha^{(2)} J_\alpha^{(2)}) \times_3 (U_\alpha^{(3)} J_\alpha^{(3)}) \quad (4)$$

is the Tucker reconstruction with adapted factors, and $s > 0$ is an optional scaling factor on the adaptation, set to $s = 1$ in all reported experiments. The individual layer weights are extracted as $\widehat{W}_\alpha^{(\ell)} = \widehat{W}_\alpha(\ell, :, :)$ for $\ell = 1, \dots, N_L$.

Remark 1 (Weight Preservation at Initialization). When $J_\alpha^{(n)} = I_{r_n}$ for all n , we have $\mathcal{T}_\alpha = \mathcal{R}_\alpha$, so $\widehat{W}_\alpha = W_\alpha$, for any value of s . The adapted model therefore starts *exactly* at the pre-trained solution, regardless of the Tucker approximation error. This property, analogous to PiSSA’s residual design, is critical for stable fine-tuning. This statement concerns the deterministic map (3)–(4) at $J_\alpha^{(n)} = I_{r_n}$. In the reported runs the $J_\alpha^{(n)}$ are initialized as $I_{r_n} + 10^{-3}E$ rather than exactly at I_{r_n} , so the model starts close to, but not exactly at, the pre-trained solution; the size of this gap is quantified in Theorem 3, and the effect of the dropout used during training in Remark 2.

Remark 2 (Implementation). The implementation stores frozen residual slices and reconstructs each adapted layer weight $\widehat{W}_\alpha^{(\ell)}$ on demand through a weight parametrization of the corresponding linear layer, avoiding materialization of the complete adapted cross-layer tensor. Dropout is applied to $J_\alpha^{(n)}$ before it multiplies $U_\alpha^{(n)}$; all reported LLaMA runs use $p = 0.005$, and dropout is inactive at evaluation time. During training, inverted dropout makes the reconstructed weights stochastic. At $J_\alpha^{(n)} = I_{r_n}$, the expected layer-wise reconstruction equals the corresponding pretrained reconstruction. Exact deterministic weight preservation holds in evaluation mode. The HOSVD factors, the core tensor and the matrices $J_\alpha^{(n)}$ are kept in float32 for numerical stability of the SVD, and the reconstructed weight is cast back to the model dtype (bfloat16 for the LLaMA experiments) in the forward pass.

3 Comparison with related work

Low-rank adaptation methods. LoRA (Hu et al., 2022) decomposes weight updates into low-rank matrices: $\Delta W = BA$. Extensions include AdaLoRA (Zhang et al., 2023) with adaptive rank allocation, QLoRA (Dettmers et al., 2023) combining quantization with low-rank adaptation, DoRA (Liu et al., 2024) decomposing weights into magnitude and direction, and GaLore (Zhao et al., 2024) projecting gradients onto low-rank subspaces. Adapter methods (Houlsby et al., 2019; Rücklé et al., 2021) insert small trainable modules between frozen layers. Despite their success, these approaches generally adapt model components on a per-layer basis and do not explicitly exploit cross-layer tensor structure.

SVD-based weight decomposition for PEFT. PiSSA (Meng et al., 2024) initializes LoRA adapters with principal singular values and vectors of pre-trained weights while freezing the residual. Related methods include OLoRA (Büyükyüz, 2024) using QR decomposition and LoRA-XS (Bałazy et al., 2024), which inserts a small trainable $r \times r$ matrix between frozen SVD-derived factors. These methods demonstrate the value of decomposing pre-trained weights but all operate *independently per layer*.

Tensor-based PEFT methods. LoTR (Bershtsky et al., 2024) applies Tucker-2 decomposition to gradient updates with shared factor matrices across layers. VeRA (Kopiczko et al., 2024) shares random frozen projection matrices across all layers while training per-layer scaling vectors. LoRETTA (Yang et al., 2024) applies tensor-train decomposition to weight updates. SuperLoRA (Chen et al., 2024) groups weight updates ΔW across layers, reshapes them into higher-order tensors, and applies Tucker decomposition. However, like LoTR, SuperLoRA decomposes the *gradient update* tensor. CRAFT is distinct: it decomposes the *pre-trained weights* using full Tucker-3 decomposition via HOSVD, preserves original weights through a residual formulation, and adapts through small trainable transformation matrices applied to each frozen factor.

Parameter complexity comparison.

Proposition 1 (Trainable Parameter Count and Scaling Comparison). *CRAFT with Tucker ranks (r_1, r_2, r_3) applied to n_p projection types has exactly $N_{\text{train}} = n_p \cdot (r_1^2 + r_2^2 + r_3^2)$ Tucker adaptation parameters (excluding the task-specific classifier head), independent of model dimension d and depth N_L for fixed ranks. For a model with N_L layers and dimension d (where $d_{\text{out}} \asymp d_{\text{in}} \asymp d$),*

the trainable parameter counts of competing methods are: **(1)** Full fine-tuning: $O(N_L \cdot d^2)$; **(2)** LoRA/PiSSA (rank r): $O(N_L \cdot r \cdot d)$; **(3)** LoTR: $O(N_L \cdot r^2 + r \cdot d)$; **(4)** CRAFT: $O(r_1^2 + r_2^2 + r_3^2)$. CRAFT is the only method among those compared whose complexity is independent of both N_L and d at fixed ranks.

Remark 3. In the GLUE experiments, task-specific classifier parameters are also trained. The 40K–41K figures reported for GLUE refer only to the CRAFT-specific Tucker adaptation parameters and exclude the task-specific classifier parameters. In the LLaMA commonsense experiments no classification head is introduced and the language-modeling head remains frozen, so the matrices $J_\alpha^{(n)}$ are the *only* trainable parameters.

4 Theoretical analysis

We provide theoretical foundations for CRAFT, including approximation guarantees, expressiveness characterization, and optimization properties. Proofs of Proposition 1, Theorem 3 and Theorem 4 are deferred to Appendix B; Theorem 1 is quoted from De Lathauwer et al. (2000) and Theorem 2 is immediate from the construction (3)–(5).

Theorem 1 (HOSVD Quasi-Optimality) (De Lathauwer et al., 2000). Let $\mathcal{W} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ with mode- n unfolding singular values $\sigma_1^{(n)} \geq \sigma_2^{(n)} \geq \dots \geq \sigma_{I_n}^{(n)}$. Let $\widetilde{\mathcal{W}}$ be its rank- (r_1, r_2, r_3) HOSVD approximation and $\mathcal{W}^*$ be the best rank- (r_1, r_2, r_3) Tucker approximation. Then

$$\|\mathcal{W} - \widetilde{\mathcal{W}}\|_F \leq \sqrt{3} \cdot \|\mathcal{W} - \mathcal{W}^*\|_F, \quad \|\mathcal{W} - \widetilde{\mathcal{W}}\|_F^2 \leq \sum_{n=1}^3 \sum_{j=r_n+1}^{I_n} (\sigma_j^{(n)})^2.$$

Moreover, if the singular values of the mode- n unfoldings decay as $\sigma_j^{(n)} = O(j^{-\beta})$ for some $\beta > 1/2$, then the squared tail satisfies $\sum_{j>r_n} (\sigma_j^{(n)})^2 = O(r_n^{1-2\beta})$, so that ranks $r_n = O(\epsilon^{-1/(\beta-1/2)})$ suffice for ϵ -approximation in the Frobenius norm.

We consider weight adaptation of the forms (3) and (4). Therefore, one can rewrite $\mathcal{T}_\alpha$ as

$$\mathcal{T}_\alpha = \widehat{\mathcal{G}}_\alpha \times_1 U_\alpha^{(1)} \times_2 U_\alpha^{(2)} \times_3 U_\alpha^{(3)}, \quad \widehat{\mathcal{G}}_\alpha = \mathcal{G}_\alpha \times_1 J_\alpha^{(1)} \times_2 J_\alpha^{(2)} \times_3 J_\alpha^{(3)}. \quad (5)$$

Consider a set $\mathcal{S}$ of orbits of tensor $\mathcal{G}_\alpha$:

$$\mathcal{S}(\mathcal{G}_\alpha) = \left\{ \mathcal{G}_\alpha \times_1 J^{(1)} \times_2 J^{(2)} \times_3 J^{(3)} : J^{(n)} \in GL(r_n) \right\}$$

where $GL(r_n)$ denotes the group of invertible $r_n \times r_n$ matrices, $n = 1, 2, 3$. Then, the collection of tensors $\mathcal{T}_\alpha$ in (5) forms a smooth manifold of dimension at most $r_1^2 + r_2^2 + r_3^2$ embedded in $\mathbb{R}^{N_L \times d_{out} \times d_{in}}$; the bound is not attained exactly because the triple $(J^{(1)}, J^{(2)}, J^{(3)})$ and $(c_1 J^{(1)}, c_2 J^{(2)}, c_3 J^{(3)})$ with $c_1 c_2 c_3 = 1$ produce the same tensor, so the stabilizer of the action is at least two-dimensional.

Theorem 2 (Orbit-Constrained Expressiveness). Let $\mathcal{W}$ have HOSVD decomposition with factors $U^{(n)}$, $n = 1, 2, 3$. Then CRAFT can adapt to any tensor $\mathcal{T}_\alpha$ of the form (5) with $\widehat{\mathcal{G}}_\alpha \in \mathcal{S}(\mathcal{G}_\alpha)$. Moreover, if $J^{(n)} = I_{r_n}$ for $n = 1, 2, 3$, then CRAFT exactly recovers the original weights: $\widehat{\mathcal{W}}_\alpha = \mathcal{W}_\alpha$.

Theorem 3 (Near-Identity Initialization Effect). Let $\mathcal{R}_\alpha$ be of the form (1) and $\mathcal{T}_\alpha^{ini} = \mathcal{G}_\alpha^{ini} \times_1 U_\alpha^{(1)} \times_2 U_\alpha^{(2)} \times_3 U_\alpha^{(3)}$. Consider initialization with $J^{(n)} = I_{r_n} + \epsilon E_{i,j}^{(n)}$ where $E_{i,j}^{(n)} \sim \mathcal{N}(0, \sigma^2)$, $\sigma = 1$, $n = 1, 2, 3$,

$$\mathcal{G}_\alpha^{ini} = \mathcal{G}_\alpha \times_1 J^{(1)} \times_2 J^{(2)} \times_3 J^{(3)}, \quad (6)$$

and let $r = \max(r_1, r_2, r_3)$. Then, for any $\delta \in (0, 1)$, with probability at least $1 - 8\delta$,

$$\|\mathcal{T}_\alpha^{ini} - \mathcal{R}_\alpha\|_F \leq C(r, \epsilon, \delta) \|\mathcal{R}_\alpha\|_F, \quad (7)$$

where $C(r, \epsilon, \delta) = O(\epsilon(\sqrt{r} + \sqrt{\log \delta^{-1}}))$ for small ϵ . Precise constants are given in Appendix B. With $\epsilon = 10^{-3}$ and the ranks used in our experiments, the right-hand side of (7) stays below $0.3 \|\mathcal{R}_\alpha\|_F$ (a worked numerical example is given in Appendix B), so the model starts in a small neighbourhood of the pre-trained solution.

4.1 Optimization properties

Theorem 4 (Gradient Flow at Initialization). *At initialization $(J^{(1)}, J^{(2)}, J^{(3)})$, the gradient of loss $\mathcal{L}$ with respect to $J^{(1)}$ is: $\frac{\partial \mathcal{L}}{\partial J^{(1)}} = U^{(1)\top} \frac{\partial \mathcal{L}}{\partial U_{\text{eff}}^{(1)}}$, where $U_{\text{eff}}^{(1)} = U^{(1)} J^{(1)}$ is the effective factor matrix. Since $U^{(1)}$ has orthonormal columns, $U^{(1)\top}$ does not amplify gradient norms:*

$$\left\| \frac{\partial \mathcal{L}}{\partial J^{(1)}} \right\|_F = \left\| U^{(1)\top} \frac{\partial \mathcal{L}}{\partial U_{\text{eff}}^{(1)}} \right\|_F \leq \left\| \frac{\partial \mathcal{L}}{\partial U_{\text{eff}}^{(1)}} \right\|_F \quad (8)$$

with equality when the gradient lies in the column space of $U^{(1)}$.

5 Experimental results

5.1 Experimental setup

We evaluate CRAFT on two sets of benchmarks: (i) the GLUE benchmark (Wang et al., 2018) using RoBERTa-base (125M parameters, 12 layers) and RoBERTa-large (355M parameters, 24 layers) (Liu et al., 2019), and (ii) eight commonsense reasoning datasets using LLaMA2-7B and LLaMA3-8B. We follow the experimental protocol from Bershatsky et al. (2024) for fair comparison on GLUE.

For RoBERTa experiments, CRAFT is applied to Q and V projections with mode-2 and mode-3 ranks $r_2 = r_3 = 100$ and mode-1 rank $r_1 = N_L$, which is the largest admissible value: $(r_1, r_2, r_3) = (24, 100, 100)$ for RoBERTa-large ($N_L = 24$), giving $2 \times (24^2 + 100^2 + 100^2) = 41,152 \approx 41\text{K}$ Tucker adaptation parameters, and $(12, 100, 100)$ for RoBERTa-base ($N_L = 12$), giving $2 \times (12^2 + 100^2 + 100^2) = 40,288 \approx 40\text{K}$. Both are reported as 0.04M in Table 2. For LLaMA experiments (both models have $N_L = 32$ layers), we use Tucker ranks $(r_1, r_2, r_3) = (32, 128, 128)$, yielding $N_{\text{train}} = 2 \times (32^2 + 128^2 + 128^2) = 67,584 \approx 68\text{K}$ Tucker adaptation parameters, reported as 0.068M throughout. We also report results with smaller ranks $(32, 64, 64)$ corresponding to $2 \times (32^2 + 64^2 + 64^2) = 18,432 \approx 18\text{K}$ parameters, with larger ranks $(32, 250, 250)$ corresponding to $252,048 \approx 0.25\text{M}$ parameters, and with the CRAFT* configuration, which adapts all four projections $\{Q, K, V, O\}$ with ranks $(32, 360, 360)$ and therefore uses $4 \times (32^2 + 360^2 + 360^2) = 1,040,896 \approx 1.0\text{M}$ parameters. Note that $r_1 = 32 = N_L$ for both LLaMA backbones, so the mode-1 factor is a full orthogonal matrix and mode 1 is not truncated in these runs; the cross-layer structure is exploited through the core tensor and the trainable $J^{(1)}$ rather than through mode-1 rank reduction.

Commonsense training protocol. We follow the LLM-Adapters protocol: training on the Commonsense-170K instruction dataset with the Alpaca-style prompt template, cross-entropy loss computed on response tokens only, and generative per-task evaluation with beam search (4 beams, at most 32 new tokens) followed by regular-expression answer extraction. We train for 3 epochs with AdamW, learning rate 3×10^{-4} , per-device batch size 4 with gradient accumulation 4 (effective batch size 16), maximum sequence length 256, warmup ratio 0.06, no weight decay, dropout $p = 0.005$ on the matrices $J_{\alpha}^{(n)}$, adaptation scaling $s = 1$, bfloat16 compute, and seed 25. We report, for each model, the epoch achieving the highest accuracy averaged over the eight tasks. In all six Q,V runs, for which complete logs are released, this was the final (third) epoch, so those numbers coincide with a fixed three-epoch protocol.

5.2 Commonsense reasoning results

Table 1 presents results on eight commonsense reasoning benchmarks. On LLaMA2-7B, CRAFT achieves 75.7 average accuracy with only 0.068M Tucker adaptation parameters significantly fewer than LoRA’s 56M parameters. These results indicate that the cross-layer

Method	# Params	BoolQ	PIQA	SIQA	Hella	Wino	ARC-e	ARC-c	OBQA	Avg.
LLaMA2-7B										
LoRA [†]	56M	69.8	79.9	79.5	83.6	82.6	79.8	64.7	81.0	77.6
PiSSA [§]	56M	67.6	78.1	78.4	76.6	78.0	75.8	60.2	75.6	73.8
MiLoRA [§]	56M	67.6	83.8	80.1	88.2	82.0	82.8	68.8	80.6	79.2
DoRA	56M	71.8	83.7	76.0	89.1	82.6	83.7	68.2	82.4	79.7
LoRA-XS [‡]	3.67M	72.1	83.7	80.5	86.0	83.9	86.0	71.2	81.0	80.5
LoRA-XS [‡]	0.92M	70.4	83.0	80.2	82.3	82.7	84.4	68.6	80.8	79.0
LoRA-XS [‡]	0.23M	67.2	81.8	78.1	75.4	80.8	81.2	65.9	74.6	75.6
CRAFT	0.018M	66.0	78.2	73.0	66.0	73.2	80.3	60.1	64.6	70.2
CRAFT	0.068M	68.3	81.1	77.3	80.1	77.5	83.5	63.9	73.6	75.7
CRAFT	0.252M	70.1	82.2	79.9	85.5	82.0	85.5	70.0	82.6	79.7
CRAFT*	1.04M	71.8	83.1	82.5	85.6	84.0	85.6	69.5	82.2	80.5
LLaMA3-8B										
LoRA [†]	57M	70.8	85.2	79.9	91.7	84.3	84.2	71.2	79.0	80.8
PiSSA [§]	57M	67.1	81.1	77.2	83.6	78.9	77.7	63.2	74.6	75.4
MiLoRA [§]	57M	68.8	86.7	77.2	92.9	85.6	86.8	75.5	81.8	81.9
DoRA	57M	74.6	89.3	79.9	95.5	85.6	90.5	80.4	85.8	85.2
LoRA-XS [‡]	3.67M	73.2	88.3	82.5	94.1	87.1	90.9	80.5	85.6	85.3
LoRA-XS [‡]	0.92M	72.8	87.4	81.5	94.2	87.4	91.3	79.2	85.0	84.8
LoRA-XS [‡]	0.23M	66.6	85.8	79.4	90.1	85.2	87.0	76.5	81.8	81.6
CRAFT	0.018M	68.5	85.7	75.8	85.7	78.3	88.9	77.9	78.2	79.9
CRAFT	0.068M	71.0	87.1	78.9	91.0	83.2	90.3	79.2	82.4	82.9
CRAFT	0.252M	73.0	87.6	80.5	93.4	87.2	91.8	80.4	85.4	84.9
CRAFT*	1.04M	73.7	88.2	82.2	94.0	86.5	92.4	80.6	86.8	85.6

All baseline results are taken from the original papers: [†]LoRA and LoRA-XS from [Bałazy et al. \(2024\)](#); ^{||}DoRA from [Liu et al. \(2024\)](#); [§]PiSSA and MiLoRA from [Wang et al. \(2025\)](#). *Q,K,V,O adapted, ranks (32, 360, 360); all other rows adapt Q,V only.

Table 1: Commonsense reasoning accuracy for fine-tuned LLaMA2-7B and LLaMA3-8B across eight datasets. Baseline numbers for LoRA, PiSSA, MiLoRA, DoRA, and LoRA-XS are reported directly from their original papers. The rank for LoRA is 32; ranks for LoRA-XS are 128, 64, and 32. DoRA uses the same rank and adapted modules as LoRA. CRAFT (ours) uses 0.018M–0.252M Tucker parameters on Q and V, independent of model dimension and depth, at Tucker ranks (32, 64, 64), (32, 128, 128) and (32, 250, 250) respectively. The highlighted CRAFT* rows additionally adapt K and O, using ~1M parameters (~2% of the 56M/57M baselines). Bold in the accuracy columns marks CRAFT* entries that attain the best value in their column within the model block; column maxima attained by a baseline are left unmarked. Bold in the # Params column simply identifies our rows.

Tucker structure is particularly effective for tasks that benefit from consistent cross-layer representations.

5.3 GLUE benchmark results

Table 2 shows that CRAFT achieves competitive performance with significantly fewer Tucker adaptation parameters:

RoBERTa-large: CRAFT achieves an 88.0 average score, coming within 0.4 points of the 3M-parameter Adpt^P adapter (88.4) while using ~75× fewer Tucker adaptation parameters. Compared to LoRA (0.8M parameters), CRAFT uses ~20× fewer parameters with only a 1.0-point lower average. On the six tasks reported by VeRA ([Kopiczko et al., 2024](#)), CRAFT averages 87.4 against VeRA’s 87.8, i.e. 0.4 points lower with 1.5× fewer parameters; we stress that CRAFT’s 88.0 figure is an eight-task average and is therefore not directly comparable to VeRA’s six-task average.

Model	Method	# Params	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
RoBERTa _{base}	FT*	125.0M	87.6	94.8	90.2	63.6	92.8	91.9	78.7	91.2	86.4
	BitFit*	0.1M	84.7	93.7	92.7	62.0	91.8	84.0	81.5	90.8	85.2
	Adpt ^D *	0.3M	87.1	94.2	88.5	60.8	93.1	90.2	71.5	89.7	84.4
	Adpt ^D *	0.9M	87.3	94.7	88.4	62.6	93.0	90.6	75.9	90.3	85.4
	LoRA*	0.3M	87.5	95.1	89.7	63.4	93.3	90.8	86.6	91.5	87.2
	VeRA [§]	0.04M	–	94.6	89.5	65.6	91.8	–	78.7	90.7	85.2 [◇]
	CRAFT (ours)	0.04M	85.7	95.1	89.2	59.0	92.0	89.0	75.8	90.4	84.5
RoBERTa _{large}	FT*	355.0M	90.2	96.4	90.9	68.0	94.7	92.2	86.6	92.4	88.9
	LoRA*	0.8M	90.6	96.2	90.9	68.2	94.9	91.6	87.4	92.6	89.0
	Adpt ^P †	3.0M	90.2	96.1	90.2	68.3	94.8	91.9	83.8	92.1	88.4
	Adpt ^P †	0.8M	90.5	96.6	89.7	67.8	94.8	91.7	80.1	91.9	87.9
	Adpt ^H †	6.0M	89.9	96.2	88.7	66.5	94.7	92.1	83.4	91.0	87.8
	Adpt ^H †	0.8M	90.3	96.3	87.7	66.3	94.7	91.5	72.9	91.5	86.4
	LoRA [†]	0.8M	90.6	96.2	90.2	68.2	94.8	91.6	85.2	92.3	88.6
	VeRA [§]	0.06M	–	96.1	90.9	68.0	94.4	–	85.9	91.7	87.8 [◇]
	CRAFT (ours)	0.04M	90.2	96.2	90.2	67.7	94.7	89.7	84.2	91.4	88.0

◇ Average computed over the 6 tasks reported by VeRA (excluding MNLI and QQP). The Avg. column for all other rows, including CRAFT, is over all eight tasks; the corresponding six-task averages for CRAFT are 83.6 (base) and 87.4 (large), quoted in the text.

Table 2: RoBERTa with different adaptation methods on the GLUE benchmark. Higher is better. * indicates numbers taken from Hu et al. (2022), including (Ben Zaken et al., 2022). † indicates numbers from Hu et al. (2022), using a restricted setup similar to Houlsby et al. (2019). § indicates numbers from Kopiczko et al. (2024), as reported in Li et al. (2024). VeRA reports on 6 GLUE tasks (excluding MNLI and QQP); we report “–” for missing tasks and compute the average over the 6 reported tasks for VeRA. CRAFT rows (highlighted) show our results with only **0.04M** Tucker adaptation parameters (excluding the task-specific classifier head).

RoBERTa-base: CRAFT achieves an 84.5 average with 0.04M Tucker adaptation parameters, compared to 87.2 for LoRA with 0.3M parameters, using $\sim 7\times$ fewer parameters. On the six-task VeRA subset CRAFT averages 83.6 against VeRA’s 85.2, so on the smaller backbone the constrained adaptation space is clearly a limitation.

Efficiency–accuracy trade-off: CRAFT occupies an extreme point on the efficiency–accuracy Pareto frontier. The method is particularly effective on larger models: on RoBERTa-large, the gap to the best baseline narrows to 1.0 point, and on LLaMA3-8B, CRAFT surpasses LoRA despite using $840\times$ fewer parameters (Table 1).

Comparison with LoTR. A direct comparison with LoTR (Bershtsky et al., 2024) in a single table is not straightforward due to different experimental protocols, and the two averages below are computed over different task subsets. Nevertheless, the results reported in Bershtsky et al. (2024) are informative: on RoBERTa-large, LoTR with 328K parameters achieves an 83.0 average, while CRAFT achieves 88.0 with only 41K parameters. These figures are quoted as published and are not a controlled comparison: the evaluated task subset, the training protocol and the hyperparameter search all differ between the two papers. These published results motivate a future controlled comparison between decomposing pre-trained weights and decomposing parameter updates.

6 Advantages, limitations, and discussion

Extreme parameter efficiency. CRAFT achieves accuracy comparable to methods with $7\text{--}840\times$ more trainable parameters. The Tucker adaptation parameter count $n_p(r_1^2 + r_2^2 + r_3^2)$ depends only on the chosen Tucker ranks and the number of adapted projection types. We emphasize that this independence holds for *fixed* Tucker ranks; whether the same

ranks suffice as models grow substantially remains an open empirical question, and the mode-1 rank is in any case constrained by $r_1 \leq N_L$. Our LLaMA experiments provide initial evidence: increasing ranks from (32, 64, 64) to (32, 128, 128) on LLaMA3-8B improved average accuracy from 79.9 to 82.9, and a further increase to (32, 250, 250) reaches 84.9. We are not aware of another PEFT method reaching comparable accuracy at this parameter budget on models of this scale.

Limitations. Further evaluation on generation tasks (e.g., instruction following, summarization) and careful hyperparameter selection would strengthen the empirical contribution. The HOSVD pre-computation adds a one-time cost dominated by the mode-2 and mode-3 unfolding SVDs, which act on $d \times (N_L d)$ matrices and therefore cost $O(N_L d^3)$. On RoBERTa-base, CRAFT shows a 2.7-point gap compared to LoRA, suggesting the constrained adaptation space may limit performance on smaller models. Results are reported for a single seed, and for the commonsense benchmarks the reported epoch is the one with the best average accuracy across the eight evaluation sets; in each of the six released Q,V runs this coincided with the final epoch, so those numbers are not inflated by epoch selection. For LLaMA2-7B at the smallest budgets, ARC-c is the weakest task relative to the baselines; increasing the Tucker adaptation budget from 0.068M to 0.252M improves ARC-c from 63.9 to 70.0 and ARC-e from 83.5 to 85.5.

7 Conclusion

We introduced CRAFT, a parameter-efficient fine-tuning method that bridges pre-trained weight decomposition with cross-layer tensor structure. The key contributions of this work are as follows:

- 1. Novel framework.** CRAFT combines frozen HOSVD factors with trainable adaptation matrices and a residual-preserving formulation that recovers the pre-trained weights exactly when the adaptation matrices equal the identity, so that training begins in a small neighbourhood of the pre-trained solution.
- 2. Extreme efficiency.** With only 40K–68K Tucker adaptation parameters at fixed ranks— independent of model dimension and depth—CRAFT is, to the best of our knowledge, the only PEFT method among those compared with this property. On LLaMA3-8B, CRAFT with 0.068M parameters achieves an 82.9 average on commonsense reasoning, surpassing LoRA (80.8) which uses $840\times$ more parameters.

Acknowledgments

The first two authors gratefully acknowledge partial support by National Science Foundation (NSF) grant DMS-2014928.

Ethics statement

This work proposes a parameter-efficient fine-tuning method. We do not foresee specific ethical concerns beyond those generally associated with fine-tuning large language models.

Use of large language models

The authors used large language model tools for assistance with text editing, citation formatting, and coding assistance during the preparation of this manuscript. All LLM-generated outputs were reviewed and verified by the authors, who take full responsibility for the content of this paper.

References

- Klaudia Bałazy, Mohammadreza Banaei, Karl Aberer, and Jacek Tabor. LoRA-XS: Low-rank adaptation with extremely small number of parameters. *arXiv preprint arXiv:2405.17604*, 2024.
- Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1–9, 2022.
- Daniel Bershtatsky, Daria Cherniuk, Talgat Daulbaev, Aleksandr Mikhalev, and Ivan Osleedets. LoTR: Low tensor rank weight adaptation. *arXiv preprint arXiv:2402.01376*, 2024.
- Kerim Büyükakyüz. OLoRA: Orthonormal low-rank adaptation of large language models. *arXiv preprint arXiv:2406.01775*, 2024.
- Xiangyu Chen, Jing Liu, Ye Wang, Pu Wang, Matthew Brand, Guanghui Wang, and Toshiaki Koike-Akino. SuperLoRA: Parameter-efficient unified adaptation for large vision models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 8050–8055, 2024.
- Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.
- Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Larousilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.
- Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.
- Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. VeRA: Vector-based random matrix adaptation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.
- Yang Li, Shaobo Han, and Shihao Ji. VB-LoRA: Extreme parameter efficient fine-tuning with vector banks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
- Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.
- Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
- Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. AdapterDrop: On the efficiency of adapters in transformers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

- Joel A. Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12:389–434, 2012.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.
- Hanqing Wang, Yixia Li, Shuo Wang, Guanhua Chen, and Yun Chen. MiLoRA: Harnessing minor singular components for parameter-efficient LLM finetuning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics (NAACL)*, pp. 4823–4836, 2025.
- Yifan Yang, Jiajun Zhou, Ngai Wong, and Zheng Zhang. LoRETTA: Low-rank economic tensor-train adaptation for ultra-low-parameter fine-tuning of large language models. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024.
- Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.
- Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

A Additional material

A.1 Choice of adapted projections

Following the original LoRA framework (Hu et al., 2022), CRAFT adapts the Q and V projection matrices while keeping K and O frozen in its main configuration. This choice is motivated by three observations:

Gauge invariance of $QK^\top$. The attention score matrix is computed as $\text{softmax}(XW_QW_K^\top X^\top / \sqrt{d_k})$. For any invertible matrix $M \in GL(d_k)$, replacing $W_Q \rightarrow W_Q M$ and $W_K \rightarrow W_K M^{-\top}$ leaves the product $W_QW_K^\top$ unchanged. This gauge invariance means that, under unconstrained perturbations, simultaneously adapting both Q and K introduces a redundant degree of freedom, motivating restriction to only one of the two.

Independence of the value pathway. The attention output for a single head is $\text{Attn}(X) = \text{softmax}(XW_QW_K^\top X^\top / \sqrt{d_k}) XW_V$. Since W_V does not enter the softmax argument, it controls an independent subspace: perturbations to W_V change *what* information is read out, while perturbations to W_Q change *where* the model attends. Adapting both Q and V therefore covers complementary degrees of freedom.

Empirical PCA analysis. PCA analysis of vectorized attention weights across ViT-B/16, RoBERTa-base, and GPT-2 (Figure 4) shows that Q weights consistently exhibit greater dispersion across layers compared to K and V, which tend to cluster near the projected mean. Specifically, for each layer ℓ , let $w_{\alpha,i}^{(\ell)} \in \mathbb{R}^{d_{\text{in}}}$ denote the i -th row of projection matrix $W_\alpha^{(\ell)}$. We pool all rows from Q, K, V, compute the sample covariance, and project onto its leading eigenvectors P_k . The *dispersion* of projection type α is

$$\sigma_\alpha^{(\ell)}(k) = \left(\frac{1}{d_{\text{out}}} \sum_{i=1}^{d_{\text{out}}} \|P_k(w_{\alpha,i}^{(\ell)} - \bar{\mu}^{(\ell)})\|^2 \right)^{1/2}. \quad (9)$$

The higher dispersion of Q weights suggests that the query projection carries the most layer-specific variation, making it the natural candidate for adaptation among Q and K .

The gauge-invariance argument above concerns the product $W_Q W_K^\top$ under *unconstrained* perturbations; under CRAFT’s constrained parametrization the redundancy is only partial, which is consistent with the observation that adapting all four projections (CRAFT*) yields a further accuracy gain in Table 1.

A.2 Design comparison of tensor-based PEFT methods

Property	LoTR	SuperLoRA	PiSSA	CRAFT
Decomposes	ΔW	ΔW	W	W
Cross-layer	Yes	Yes	No	Yes
Tucker type	Tucker-2	Tucker- n	N/A	Tucker-3
What trains	Core slices	Factors	A, B	$J^{(1)}, J^{(2)}, J^{(3)}$
Factors frozen	No	No	No	Yes
Residual	No	No	Yes	Yes
Indep. d, N_L	No	No	No	Yes

Table 3: Design comparison of tensor-based PEFT methods. “Indep. d, N_L ” indicates whether the Tucker adaptation parameter count is independent of model dimension d and number of layers N_L at fixed ranks.

A.3 PEFT method taxonomy

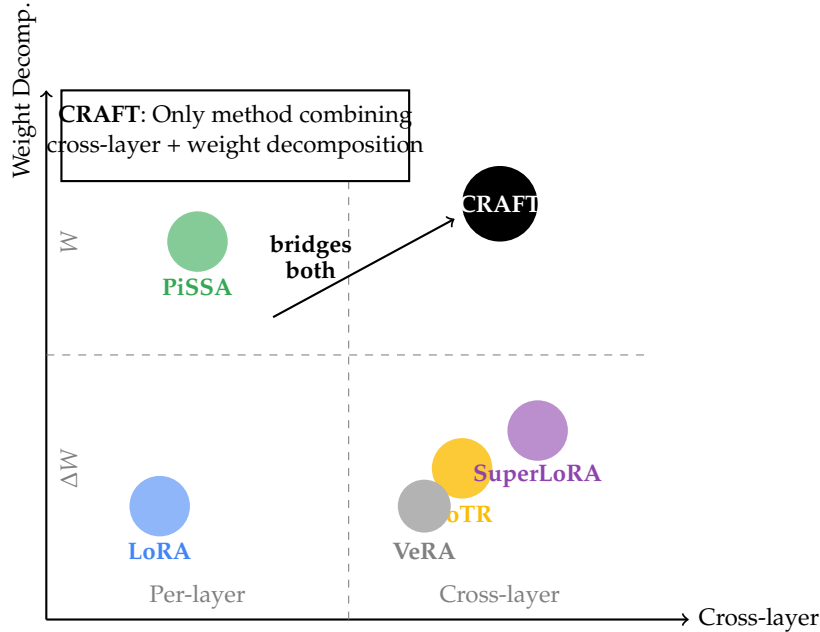


Figure 2: PEFT method taxonomy. CRAFT uniquely combines cross-layer tensor structure with pre-trained weight decomposition, among the methods compared here.

A.4 Parameter scaling analysis

Figure 3 illustrates the parameter-scaling behavior described in Proposition 1.

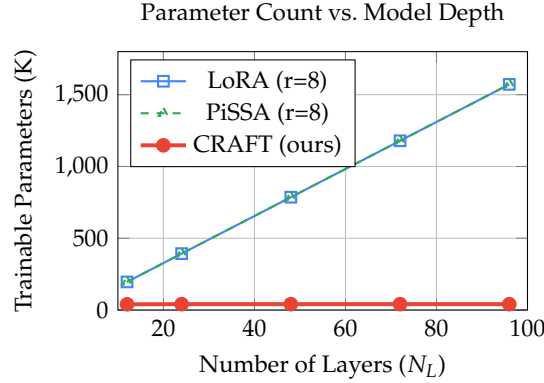


Figure 3: Parameter count vs. model depth. At fixed Tucker ranks, CRAFT’s Tucker adaptation parameter count remains **constant** regardless of model depth, while LoRA and PiSSA scale linearly with N_L . The LoRA/PiSSA curves assume $d = 1024$, rank $r = 8$ and one adapted projection per layer ($2rd = 16,384$ parameters per layer); the CRAFT curve corresponds to ranks $(\min(N_L, 24), 100, 100)$ on two projection types, i.e. 40K at $N_L = 12$ and 41K for $N_L \geq 24$, since the mode-1 rank is capped by $r_1 \leq N_L$. Counting LoRA on two projections would double its slope, so the comparison shown is conservative. Whether the same ranks suffice for significantly deeper models is an open question (see Section 6).

A.5 PCA analysis of attention weights

Figure 4 visualizes the attention-weight PCA analysis used to motivate the choice of adapted projections.

B Proofs

B.1 Proof of Proposition 1

For the sake of justification of the parameter counts above, observe that full fine-tuning trains all N_L weight matrices of size $d \times d$, giving $O(N_L d^2)$. LoRA and PiSSA introduce two rank- r factors per layer of sizes $d \times r$ and $r \times d$, yielding $O(N_L r d)$ parameters. LoTR shares factor matrices of size $d \times r$ across layers but maintains per-layer core slices of size $r \times r$, giving a parameter count of $O(N_L r^2 + r d)$.

For each projection type α , CRAFT’s trainable parameters are $J_\alpha^{(n)} \in \mathbb{R}^{r_n \times r_n}$, $n = 1, 2, 3$, which results in $n_p(r_1^2 + r_2^2 + r_3^2)$ parameters independent of d and N_L . Note that the constraint $r_1 \leq N_L$ must hold for the mode-1 factor to be well defined; the count is independent of N_L for any fixed admissible r_1 .

B.2 Proof of Theorem 3

Observe that

$$\|\mathcal{T}_\alpha^{\text{ini}} - \mathcal{R}_\alpha\|_F = \|\mathcal{G}_\alpha^{\text{ini}} - \mathcal{G}_\alpha\|_F = \|g_\alpha^{\text{ini}} - g_\alpha\|,$$

where the first equality uses that the factor matrices $U_\alpha^{(n)}$ have orthonormal columns, and where $g_\alpha^{\text{ini}}, g_\alpha \in \mathbb{R}^{r_1 r_2 r_3}$ are the vectorized versions of the tensors $\mathcal{G}_\alpha^{\text{ini}}, \mathcal{G}_\alpha$. Note that, by (6), one has

$$g_\alpha^{\text{ini}} = (J^{(3)} \otimes J^{(2)} \otimes J^{(1)}) g_\alpha.$$

Then, since $\|g_\alpha\| = \|\mathcal{G}_\alpha\|_F = \|\mathcal{R}_\alpha\|_F$, we derive

$$\|g_\alpha^{\text{ini}} - g_\alpha\| \leq C(r, \epsilon, \delta) \|\mathcal{R}_\alpha\|_F, \quad (10)$$

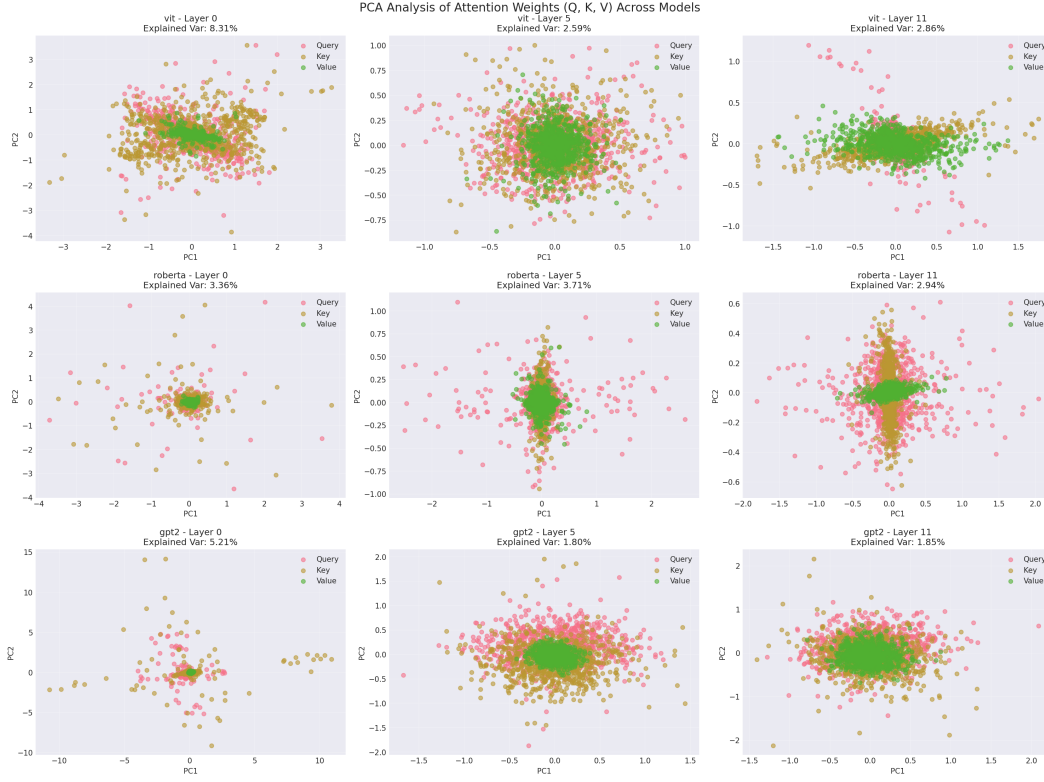


Figure 4: PCA of vectorized attention weights across models and layers. Each point represents one row of a weight matrix $W_{\alpha}^{(\ell)}$, projected onto the first two principal components computed from the pooled set $\{w_{\alpha}^{(\ell)}\}$ at each layer. Q (pink) exhibits higher dispersion (Eq. 9), while K (olive) and V (green) concentrate near the origin. The two-component explained-variance ratio (shown in titles) decreases in deeper layers for ViT and GPT-2, indicating that the weight distribution spreads over a higher effective dimensionality.

where

$$C(r, \epsilon, \delta) = \sigma_{\max} \left(J^{(3)} \otimes J^{(2)} \otimes J^{(1)} - I_{r_1} \otimes I_{r_2} \otimes I_{r_3} \right).$$

Here, $\sigma_{\max}(A)$ denotes the largest singular value of a matrix A . Substituting $J^{(n)} = I_{r_n} + \epsilon E^{(n)}$ and expanding the Kronecker product gives

$$J^{(3)} \otimes J^{(2)} \otimes J^{(1)} - I_{r_3} \otimes I_{r_2} \otimes I_{r_1} = \epsilon \sum_{n=1}^3 A_n + \epsilon^2 \sum_{1 \leq m < n \leq 3} A_m A_n + \epsilon^3 A_1 A_2 A_3,$$

where $A_1 = I_{r_3} \otimes I_{r_2} \otimes E^{(1)}$, $A_2 = I_{r_3} \otimes E^{(2)} \otimes I_{r_1}$ and $A_3 = E^{(3)} \otimes I_{r_2} \otimes I_{r_1}$ all act on $\mathbb{R}^{r_1 r_2 r_3}$. Since $\sigma_{\max}(A \otimes B) = \sigma_{\max}(A) \sigma_{\max}(B)$, one has $\sigma_{\max}(A_n) = \sigma_{\max}(E^{(n)})$, and the triangle inequality together with submultiplicativity yields

$$\begin{aligned} C(r, \epsilon, \delta) &\leq \epsilon [\sigma_{\max}(E^{(1)}) + \sigma_{\max}(E^{(2)}) + \sigma_{\max}(E^{(3)})] \\ &\quad + \epsilon^2 [\sigma_{\max}(E^{(1)}) \sigma_{\max}(E^{(2)}) + \sigma_{\max}(E^{(1)}) \sigma_{\max}(E^{(3)}) + \sigma_{\max}(E^{(2)}) \sigma_{\max}(E^{(3)})] \\ &\quad + \epsilon^3 \sigma_{\max}(E^{(1)}) \sigma_{\max}(E^{(2)}) \sigma_{\max}(E^{(3)}). \end{aligned} \quad (11)$$

It is well known (Tropp, 2012) that, for any $\delta \in (0, 1)$ and $k = 1, 2, 3$,

$$\mathbb{P} \left\{ \sigma_{\max}(E^{(k)}) \leq 2\sqrt{r_k} + \sqrt{2 \log(\delta^{-1})} \right\} \geq 1 - \delta. \quad (12)$$

Combining (11) and (12) with the union bound yields

$$\begin{aligned} C(r, \epsilon, \delta) &\leq 3 \left[\epsilon \left(2\sqrt{r} + \sqrt{2\log(\delta^{-1})} \right) \right] \\ &+ 3 \left[\epsilon \left(2\sqrt{r} + \sqrt{2\log(\delta^{-1})} \right) \right]^2 + \left[\epsilon \left(2\sqrt{r} + \sqrt{2\log(\delta^{-1})} \right) \right]^3. \end{aligned} \quad (13)$$

Hence, if $\epsilon \left(2\sqrt{r} + \sqrt{2\log(\delta^{-1})} \right) < 1$, then, with probability at least $1 - 8\delta$,

$$\left\| \mathcal{T}_\alpha^{\text{ini}} - \mathcal{R}_\alpha \right\|_F \leq 7\epsilon \left(2\sqrt{r} + \sqrt{2\log(\delta^{-1})} \right) \left\| \mathcal{R}_\alpha \right\|_F, \quad (14)$$

so that the adapted model is initialized in close proximity to the pre-trained solution. For instance, with $\epsilon = 10^{-3}$, $r = 128$ and $\delta = 10^{-3}$, the right-hand side of (14) equals approximately $0.18 \left\| \mathcal{R}_\alpha \right\|_F$, and the condition $\epsilon(2\sqrt{r} + \sqrt{2\log \delta^{-1}}) < 1$ is comfortably satisfied. At the largest rank used in this paper, $r = 360$, the same bound gives $0.29 \left\| \mathcal{R}_\alpha \right\|_F$.

B.3 Proof of Theorem 4

By the chain rule:

$$\frac{\partial \mathcal{L}}{\partial J_{ij}^{(1)}} = \sum_{k,l} \frac{\partial \mathcal{L}}{\partial (U_{\text{eff}}^{(1)})_{kl}} \frac{\partial (U_{\text{eff}}^{(1)})_{kl}}{\partial J_{ij}^{(1)}}.$$

Since $(U_{\text{eff}}^{(1)})_{kl} = \sum_m U_{km}^{(1)} J_{ml}^{(1)}$:

$$\frac{\partial (U_{\text{eff}}^{(1)})_{kl}}{\partial J_{ij}^{(1)}} = U_{ki}^{(1)} \delta_{jl}$$

Therefore:

$$\frac{\partial \mathcal{L}}{\partial J_{ij}^{(1)}} = \sum_k U_{ki}^{(1)} \frac{\partial \mathcal{L}}{\partial (U_{\text{eff}}^{(1)})_{kj}} = \left(U^{(1)\top} \frac{\partial \mathcal{L}}{\partial U_{\text{eff}}^{(1)}} \right)_{ij}$$