Finer Parameter Steps for Low-Rank PEFT: A Controlled Study with CP Tensor Adapters

Xinjue Wang [1] **Xiuheng Wang** [2] **Yejun Zhang** [3] **Sergiy A. Vorobyov** [1] **Esa Ollila** [1] **Zhi-Yong Wang** [4]

Abstract

Low-rank adapters are usually compared by sweeping a small set of ranks, but the rank also fixes the resolution of the parameter budget. For a 2048×2048 OPT attention projection, increasing LoRA by one rank stores 4096 trainable scalars, leaving large gaps between feasible low-budget adapter sizes. This paper asks whether a tensorized adapter with finer capacity increments changes the observed accuracy–budget trade-off. We instantiate this question with fixed-component canonical polyadic (CP) tensor adapters. Under a $32\times64\times32\times64$ tensorization, one normalized CP component stores 193 trainable scalars per projection, about 21 times smaller than one LoRA rank step. We compare CP adapters and LoRA on OPT-1.3B across SST-2, RTE, and BoolQ under matched target modules, training protocol, data caps, and seed schedules. CP trains stably and fills the gaps between LoRA ranks, but the effect is task-dependent: SST-2 reaches an early low-budget plateau, BoolQ benefits from additional CP components before saturating slightly below LoRA, and RTE remains LoRA-favored. Finer parameter steps are therefore useful for diagnosing PEFT budget sensitivity, but they do not by themselves guarantee a better accuracy–budget curve.

[1]Department of Information and Communications Engineering, Aalto University, 02150, Espoo, Finland [2]School of Future Technology, South China University of Technology, Guangzhou, 511442, China [3]Department of Computer Science, Aalto University, 02150, Espoo, Finland [4]State Key Laboratory of Ocean Sensing, Ocean College, Zhejiang University, Hangzhou 310000, China. Correspondence to: Xiuheng Wang <dr.xiuheng.wang@gmail.com>.

1. Introduction

Parameter-efficient fine-tuning (PEFT) adapts a frozen pre-trained model by training a small number of additional parameters (Hu et al., 2022; Liu et al., 2024; Yang et al., 2024a). LoRA is a standard PEFT baseline because its update $\Delta W = BA$ is simple, mergeable, and effective in practice (Hu et al., 2022). The rank r controls both the expressivity and the trainable-parameter count of the update. This convention is natural, but it also means that the rank fixes the resolution at which the accuracy–budget curve can be observed. For a large projection matrix, even one rank increment can add thousands of trainable scalars, so the low-budget region may be only sparsely sampled.

Tensorized adapters provide a way to change this budget grid. After reshaping a matrix update into a multiway tensor, a canonical polyadic (CP) adapter can add capacity one component at a time. For the OPT-1.3B attention projections studied here, one LoRA rank step stores 4096 scalars per target projection, whereas one normalized CP component under our main tensorization stores 193. This finer grid could reveal useful operating points between LoRA ranks. At the same time, CP components impose Kronecker-structured directions after reshaping, so finer budget resolution may come with reduced update flexibility.

This trade-off motivates a simple question: when two PEFT families have different parameter-step sizes, should they be compared only at matched budgets, or through the best accuracy each can attain under a budget cap? We study this question with a best-under-budget comparison: for each budget cap B, we report the best tested accuracy from all settings whose trainable-parameter count is at most B. This view makes the discretization of feasible budgets explicit, while avoiding the claim that parameter arithmetic alone predicts accuracy.

We evaluate this lens on OPT-1.3B with three classification tasks: SST-2, RTE, and BoolQ. The target modules, trainer, data caps, checkpoint-selection rule, and seed schedules are held fixed across methods. Matched-budget runs test whether fixed-component CP adapters are viable at LoRA-scale budgets, and a denser CP sweep probes the intervals between LoRA ranks. Figure 1 illustrates the resulting

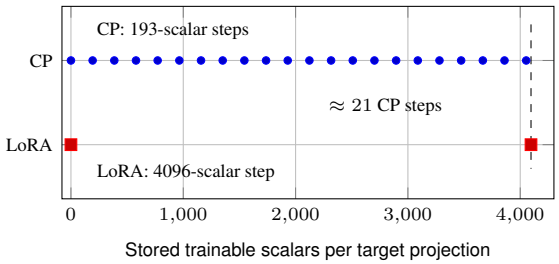

Figure 1. Discrete budget grids for a single 2048×2048 target projection. Under the $2048 = 32 \times 64$ row/column split, a LoRA rank step stores 4096 trainable scalars, while a normalized CP component stores 193.

budget grid for one target projection.

The answer is task-dependent rather than uniformly positive. On SST-2, several very small CP adapters reach an early plateau near 0.93 accuracy, but no tested CP setting exceeds the best LoRA result under the same budget caps. On BoolQ, additional CP components matter in the low-budget region before the curve saturates slightly below LoRA. On RTE, LoRA remains stronger after selected ten-seed confirmation. These results suggest that finer parameter steps are useful as a diagnostic for PEFT budget sensitivity: they reveal where budget matters, but they do not automatically overcome an expressivity gap.

Our contributions are:

1. We make discrete budget steps explicit by comparing PEFT families through the best tested setting under each budget cap.
2. We instantiate this lens with fixed-component CP tensor adapters and LoRA under a controlled OPT-1.3B protocol.
3. We show that finer parameter steps reveal task-dependent budget-response patterns: early saturation on SST-2, gradual saturation on BoolQ, and a persistent LoRA advantage on RTE.

We do not claim that the fixed CP adapter studied here is a new state-of-the-art PEFT method. Instead, we use it as a controlled tensorized adapter family whose capacity parameter has a much smaller step size than LoRA, allowing us to isolate how budget granularity changes the observed accuracy–budget curve.

2. Related Work

LoRA variants and rank allocation. LoRA (Hu et al., 2022) established low-rank adaptation as a practical default for PEFT. Subsequent work modifies the update parameterization or decides how much rank to allocate, including adaptive budget allocation (Zhang et al., 2023), weight-decomposed adaptation (Liu et al., 2024), autonomous rank growth (Sheng et al., 2025), and non-zero initialization (Li et al., 2025). Surveys organize these variants across architecture, optimization, and deployment axes (Yang et al., 2024a; Li et al., 2026). Fixed-rank and Riemannian formulations study complementary geometric aspects of low-rank optimization (Bian et al., 2025; Zhang & Pilanci, 2024). Our focus is different: we isolate the budget resolution induced by the discrete capacity parameter itself.

Tensorized and structured adapters. Tensor decompositions provide classical tools for representing multiway structure (Kolda & Bader, 2009). Recent tensorized PEFT methods replace dense low-rank factors with structured tensor parameterizations, including tensor-train formulations such as LoRETTA (Yang et al., 2024c), CP-style adaptations such as LoRTA (Hounie et al., 2024) and CaRA (Veeramacheneni et al., 2025), and broader structured-update families such as TensLoRA (Marmoret et al., 2025), AdaZeta (Yang et al., 2024b), TeRA (Gu et al., 2025), and KRAdapter (Albert et al., 2025). These methods often combine parameter savings with choices about sharing, tensorization, initialization, or adaptive allocation. We use a fixed-component CP adapter as a controlled test case rather than as a claim about the best tensor-adapter design.

Budget-aware comparison. PEFT methods are often compared at a few matched parameter counts or ranks. Such comparisons are necessary, but they can hide the fact that different adapter families offer different feasible budget grids. Our budget-cap view is complementary: it asks what each family can achieve over the tested grid, making budget discretization visible in the evaluation itself.

3. Comparing Parameter Steps Under Budget Caps

Let $\mathcal{A}$ be an adapter family with a discrete capacity parameter k and stored trainable-parameter count $P_{\mathcal{A}}(k)$. We define its *parameter step* as

$$\Delta P_{\mathcal{A}}(k) = P_{\mathcal{A}}(k+1) - P_{\mathcal{A}}(k). \tag{1}$$

For LoRA on a single $m \times n$ target matrix, $P_{\text{LoRA}}(r) = r(m+n)$, so

$$\Delta P_{\text{LoRA}} = m + n. \tag{2}$$

For OPT-1.3B attention projections, $m = n = 2048$, giving $\Delta P_{\text{LoRA}} = 4096$ stored trainable scalars per projection.

The parameter step is an accounting quantity, not an accuracy predictor. It specifies the spacing of feasible budgets for a family. To compare families under a budget cap, we

report the observed best-under-budget curve

$$U_{\mathcal{A}}(B) = \max_{k \in \mathcal{K}_{\mathcal{A}}:\, P_{\mathcal{A}}(k) \le B} A_{\mathcal{A}}(k), \tag{3}$$

where $\mathcal{K}_{\mathcal{A}}$ is the tested capacity grid and $A_{\mathcal{A}}(k)$ is held-out evaluation accuracy after best-development checkpoint selection. Thus $U_{\mathcal{A}}(B)$ describes the best tested setting whose trainable-parameter count does not exceed B.

This curve is descriptive rather than a deployment model-selection protocol. Because CP has more tested capacity values than LoRA, small differences in the best-under-budget curve should not be over-interpreted as reliable wins. We therefore report the raw curves, matched-budget comparisons, and selected ten-seed confirmations together with the best-under-budget summary.

We report both trainable parameter count and adapter Adam-state memory. In this setup the two are proportional: one LoRA rank step adds $196{,}608$ trainable parameters across the model and about 1.50 MB of Adam state, while one CP component step adds $9{,}264$ trainable parameters and about 0.071 MB. This memory accounting concerns adapter optimizer state; it is not a measure of total training memory, which also includes the frozen backbone, activations, and implementation-specific overhead.

For LoRA, the tested ranks $\{1, 2, 3, 4, 6, 8\}$ give six budget levels. The gaps between them are large: for instance, there is no tested LoRA point between $r = 1$ (196k parameters) and $r = 2$ (393k parameters). A family with smaller parameter steps can place tested settings inside these gaps. This comparison asks whether those extra settings change the best observed accuracy under the same budget cap.

4. CP Tensor Adapters

We use normalized CP tensor adapters as a fixed-component adapter family with smaller parameter steps than LoRA. All main runs use standard first-order optimization and fix the component count before training. We call the adapters fixed-component because the value of c is chosen before each run; the experiments then sweep this fixed value across runs.

Adapter form. For each target matrix $W_0 \in \mathbb{R}^{m \times n}$, we train

$$W = W_0 + \Delta W, \tag{4}$$

with W_0 frozen. This is adaptation, not compression.

Row/column tensorization. We reshape the matrix update by splitting both the row index and the column index. For the main split, a row index of length 2048 is written as two indices of sizes 32 and 64; the column index is split in the same way. A 2048×2048 update is therefore viewed as a four-way tensor,

$$\mathcal{T}(\Delta W) \in \mathbb{R}^{32 \times 64 \times 32 \times 64}. \tag{5}$$

An entry $\Delta W_{a,b}$ is indexed as $\mathcal{T}(\Delta W)_{a_1,a_2,b_1,b_2}$, where (a_1, a_2) identifies the original row and (b_1, b_2) identifies the original column. This reshape is not optimized; Table 5 reports a small sensitivity check over alternative splits. Each selected projection is tensorized independently, without cross-layer sharing or adaptive component allocation.

Normalized CP parameterization. The tensorized update is a sum of c rank-one components with normalized factors:

$$\mathcal{T}(\Delta W) = \sum_{s=1}^{c} \lambda_s u_s^{(1)} \circ u_s^{(2)} \circ u_s^{(3)} \circ u_s^{(4)}, \qquad \|u_s^{(\ell)}\|_2 = 1. \tag{6}$$

Each factor $u_s^{(1)} \in \mathbb{R}^{32}$, $u_s^{(2)} \in \mathbb{R}^{64}$, $u_s^{(3)} \in \mathbb{R}^{32}$, and $u_s^{(4)} \in \mathbb{R}^{64}$ has unit norm. The scalar λ_s carries the component scale. In our implementation, the optimizer stores unconstrained raw factors and normalizes them in the forward pass. This removes factor-scale ambiguity and was sufficient for stable first-order training. We count stored trainable scalars rather than the intrinsic degrees of freedom implied by the unit-norm constraints.

Per-component parameter cost. One CP component under this tensorization stores $32 + 64 + 32 + 64$ factor entries plus one scalar amplitude, or 193 trainable scalars per target projection. By contrast, one LoRA rank step stores 4096 scalars for the same projection. After reshaping back to a matrix, a CP component corresponds, up to the index ordering, to

$$\Delta W_s = \lambda_s \big(u_s^{(1)} \otimes u_s^{(2)}\big)\big(u_s^{(3)} \otimes u_s^{(4)}\big)^{\top}. \tag{7}$$

A c-component CP adapter therefore gives a matrix update of rank at most c, but its rank-one directions are Kronecker-structured rather than arbitrary dense outer products. The smaller parameter count comes from this restriction. Across the 48 adapted projections in OPT-1.3B, one LoRA rank adds $196{,}608$ trainable parameters and about 1.50 MB of Adam optimizer state. One CP component adds $9{,}264$ trainable parameters and about 0.071 MB. Thus a single LoRA rank step has roughly the same adapter budget as 21 CP component steps.

Why fix the component count? We fix c in advance rather than growing it during training. This choice isolates parameter-step size from adaptive allocation. A growing-component rule could be useful, but it would introduce an additional design variable and obscure the budget-resolution question studied here.

Table 1. Stored trainable parameters and Adam optimizer-state footprint over all adapted target projections (48 projections: `q_proj` and `v_proj` in all 24 OPT decoder layers).

Budget	Method	Setting	Params	Opt. MB
Tiny	LoRA	$r = 1$	196,608	1.500
Tiny	CP	$c = 21$	194,544	1.484
Low	LoRA	$r = 2$	393,216	3.000
Low	CP	$c = 43$	398,352	3.039
Mid	LoRA	$r = 4$	786,432	6.000
Mid	CP	$c = 85$	787,440	6.008
High	LoRA	$r = 8$	1,572,864	12.000
High	CP	$c = 171$	1,584,144	12.086

5. Experiments

5.1. Protocol

All experiments use `facebook/opt-1.3b`, adapt the `q_proj` and `v_proj` attention modules (48 target projections), and train with a standard HuggingFace Trainer. We cap each task at 1000 training, 500 development, and 1000 held-out evaluation examples. Realized evaluation counts are 872 for SST-2, 277 for RTE, and 1000 for BoolQ. All runs use 5000 training steps with evaluation every 1000 steps, best-development checkpoint selection, and an fp16 backbone. The base grid uses seeds $0, 1, 2$; selected confirmation cells use seeds 0–9 under the same protocol. LoRA uses learning rate 10^{-4} and CP uses 2×10^{-4}, chosen before the main grid. We did not perform a full per-method learning-rate sweep, so the comparison should be read as a controlled pilot rather than a fully optimized benchmark.

5.2. Matched-budget comparison

CP trains at LoRA-scale budgets. Matched budgets, however, do not imply matched behavior. Table 2 reports matched-budget results averaged over seeds $0, 1, 2$. CP trains stably in every cell and reaches evaluation accuracy comparable to LoRA on SST-2 and BoolQ. On RTE, LoRA is consistently stronger. The matched-budget comparison supports CP viability under this protocol, while also showing that equal parameter counts do not remove method-dependent differences.

5.3. Best under a budget cap

Finer CP steps reveal three regimes. The main sweep evaluates budgets between the tested LoRA ranks. We use LoRA ranks $r \in \{1, 2, 3, 4, 6, 8\}$ and CP component counts $c \in \{1, 2, 4, 8, 16, 21, 28, 36, 43, 64, 85, 128, 171\}$. Figure 2 shows the raw accuracy–budget curves with one-standard-deviation error bars over all available seeds. Table 3 reports the descriptive best-under-budget curves defined by Eq. (3). The horizontal axis in Figure 2 is trainable parameters, but it can also be read as adapter optimizer-state memory because the two are proportional here. For example, LoRA $r = 1$ and CP $c = 21$ both use about 1.5 MB of Adam state, while the CP points $c = 1, 2, 4, 8, 16$ all sit before the first LoRA rank. Selected cells were extended to ten seeds to verify the SST-2 low-budget plateau, BoolQ rise-and-saturation, and RTE persistent-gap patterns suggested by the base grid.

The raw curves are non-monotone, but three task-level patterns are visible.

SST-2: early plateau. Small CP settings reach accuracy around 0.93 with a small fraction of the LoRA $r = 1$ budget, and more components do not improve the curve monotonically. The three-seed grid places the best early CP point at $c = 2$ (0.934 ± 0.009), but ten-seed checks at $c = 4$ and $c = 21$ show lower means (0.929 ± 0.006 and 0.930 ± 0.005). We therefore interpret the result as a low-budget plateau rather than as evidence that $c = 2$ is a reliable optimum. No tested CP point exceeds the best LoRA result under the same budget caps on SST-2. The useful observation is that much of the low-budget region is already close to saturated, and LoRA's sparse rank grid does not resolve that region.

BoolQ: rise and saturation. The CP curve rises from 0.662 at $c = 1$ to 0.714 at $c = 4$, continues improving to 0.737 ± 0.012 at $c = 43$, and then saturates near 0.739 at $c = 64$. The region from $c = 1$ to $c = 43$ is the informative part of the CP sweep. Beyond that, adding more components gives little. The best LoRA result under these budget caps remains slightly higher in mean, with ten-seed LoRA $r = 1$ at 0.743 ± 0.013.

RTE: persistent gap. After selected ten-seed confirmation, CP remains below the best LoRA result under the same budget caps. The confirmed CP means are 0.738 ± 0.030 at $c = 28$ and 0.736 ± 0.013 at $c = 64$, while LoRA $r = 6$ reaches 0.760 ± 0.015. Under this fixed tensorization and protocol, finer CP steps and additional components do not close the gap. The higher CP mean at $c = 28$ also comes with larger variance, so we do not read it as a precise ordering between the two CP settings.

Table 4 reports the selected ten-seed confirmations. These cells verify the qualitative SST-2 plateau, BoolQ rise-and-saturation, and RTE persistent-gap patterns suggested by the base grid.

Figure 3 zooms into the low-budget region ($c \leq 43$, LoRA $r = 1, 2$). This isolates the budget range before and around the first two LoRA ranks. The first LoRA point is $r = 1$, while CP has multiple tested component counts before and around the same memory level. SST-2 already has several CP points around 0.93 inside the first LoRA rank interval. BoolQ rises more gradually and remains below LoRA $r = 1$. RTE has higher variance and no clear low-budget improvement trend.

Table 2. Matched-budget results averaged over seeds $0, 1, 2$. Δ eval is CP minus LoRA.

Task	Budget	LoRA dev	LoRA eval	CP dev	CP eval	Δ eval
SST2	Low	0.931	0.937	0.925	0.931	-0.006
SST2	Mid	0.931	0.939	0.925	0.933	-0.005
SST2	High	0.935	0.932	0.923	0.936	+0.004
RTE	Low	0.768	0.747	0.770	0.732	-0.016
RTE	Mid	0.789	0.753	0.762	0.722	-0.031
RTE	High	0.771	0.745	0.760	0.729	-0.016
BoolQ	Low	0.761	0.741	0.744	0.735	-0.006
BoolQ	Mid	0.753	0.742	0.751	0.740	-0.001
BoolQ	High	0.756	0.735	0.754	0.740	+0.005

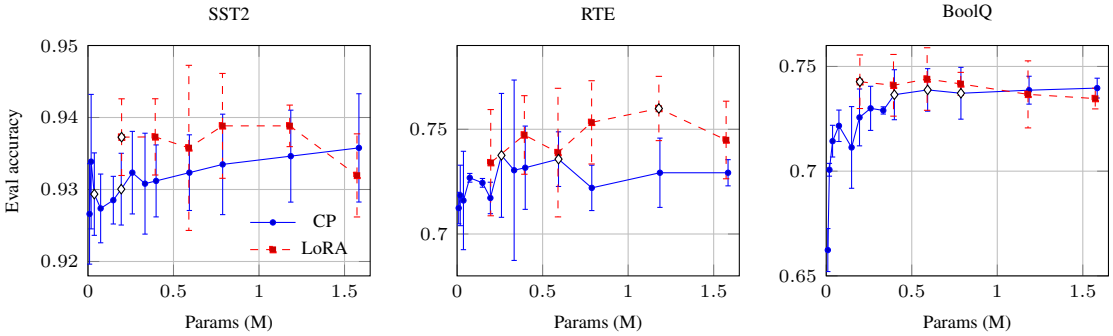

Figure 2. Raw accuracy–budget curves with one-standard-deviation error bars over all available seeds. The x-axis is trainable parameters in millions; the same ordering also gives adapter Adam-state memory because each trainable parameter has the same optimizer state. Axis ranges are task-specific to show low-budget variation. Most grid points use seeds $0, 1, 2$; diamond markers indicate selected cells confirmed with seeds 0–9.

5.4. Tensorization sensitivity

The tested reshapes give similar SST-2 results. The CP step size depends on how the row and column indices are factorized. Table 5 compares three tensorizations at the SST-2 mid-budget level, with CP component counts adjusted to approximately match the trainable-parameter budget.

All three tensorizations train stably and remain within a narrow evaluation band. The $16 \times 16 \times 8$ split is strongest in this small study, but the spread is modest. The reshape affects both step size and component structure, yet the tested alternatives do not indicate brittleness on SST-2. We therefore treat the 32×64 split as a representative setting rather than an optimized tensorization.

6. Discussion

6.1. Finer steps as a diagnostic

Finer parameter steps are most useful when the accuracy–budget curve changes inside a LoRA rank interval. BoolQ is the clearest example: the CP sweep reveals a gradual rise from very small adapters to the mid-budget region before the curve saturates. SST-2 shows the opposite pattern, with several low-budget CP points already near the task's plateau under this protocol. RTE suggests that budget granularity is not the main bottleneck. Thus the value of finer steps is diagnostic: they reveal the shape of the budget-response curve, even when they do not improve the best-under-budget result.

6.2. Granularity versus expressivity

The experiments separate budget granularity from update expressivity. CP provides many more feasible budgets than LoRA in the same parameter range, but each CP rank-one direction is Kronecker-structured after tensorization. LoRA rank increments are coarser, yet each increment adds an unconstrained dense rank-one direction. The RTE gap suggests that, for some tasks, this flexibility matters more than a finer budget grid. The BoolQ curve suggests the complementary case: a finer grid can reveal useful intermediate capacity growth before saturation, even if the final mean remains slightly below LoRA.

6.3. Semantically grounded tensorization

The tensorization used here is imposed on language-model matrices rather than derived from semantic tensor modes. Domains with native multiway structure may provide a more direct testbed for tensorized adapters. Wireless communication is one example, where signal and channel objects

Table 3. Best tested accuracy under each LoRA budget. For $B = P_{\mathrm{LoRA}}(r)$, $r^\star$ and $c^\star$ denote the best tested LoRA and CP settings not exceeding B. CP budget fraction is the CP parameter count divided by B; gap is $U_{CP}(B) - U_{\mathrm{LoRA}}(B)$. † entries use seeds 0–9; all other entries use seeds $0, 1, 2$. The comparison is descriptive over the tested grid.

Task	Budget	$r^\star$	$U_{\mathrm{LoRA}}(B)$	$c^\star$	$U_{CP}(B)$	CP budget fraction	Gap
SST2	$r = 1$	1	$0.937 \pm 0.005^\dagger$	$c = 2$	0.934 ± 0.009	9.4%	-0.003
SST2	$r = 2$	2	0.937 ± 0.005	$c = 2$	0.934 ± 0.009	4.7%	-0.003
SST2	$r = 3$	2	0.937 ± 0.005	$c = 2$	0.934 ± 0.009	3.1%	-0.003
SST2	$r = 4$	4	0.939 ± 0.007	$c = 2$	0.934 ± 0.009	2.4%	-0.005
SST2	$r = 6$	4	0.939 ± 0.007	$c = 2$	0.934 ± 0.009	1.6%	-0.005
SST2	$r = 8$	4	0.939 ± 0.007	$c = 128$	0.935 ± 0.006	75.4%	-0.004
RTE	$r = 1$	1	0.734 ± 0.025	$c = 8$	0.727 ± 0.002	37.7%	-0.007
RTE	$r = 2$	2	0.747 ± 0.019	$c = 28$	$0.738 \pm 0.030^\dagger$	66.0%	-0.010
RTE	$r = 3$	2	0.747 ± 0.019	$c = 28$	$0.738 \pm 0.030^\dagger$	44.0%	-0.010
RTE	$r = 4$	4	0.753 ± 0.020	$c = 28$	$0.738 \pm 0.030^\dagger$	33.0%	-0.016
RTE	$r = 6$	6	$0.760 \pm 0.015^\dagger$	$c = 28$	$0.738 \pm 0.030^\dagger$	22.0%	-0.022
RTE	$r = 8$	6	$0.760 \pm 0.015^\dagger$	$c = 28$	$0.738 \pm 0.030^\dagger$	16.5%	-0.022
BoolQ	$r = 1$	1	$0.743 \pm 0.013^\dagger$	$c = 21$	0.726 ± 0.014	99.0%	-0.017
BoolQ	$r = 2$	1	$0.743 \pm 0.013^\dagger$	$c = 28$	0.730 ± 0.011	66.0%	-0.013
BoolQ	$r = 3$	3	0.744 ± 0.015	$c = 43$	$0.737 \pm 0.012^\dagger$	67.5%	-0.007
BoolQ	$r = 4$	3	0.744 ± 0.015	$c = 64$	$0.739 \pm 0.010^\dagger$	75.4%	-0.005
BoolQ	$r = 6$	3	0.744 ± 0.015	$c = 64$	$0.739 \pm 0.010^\dagger$	50.3%	-0.005
BoolQ	$r = 8$	3	0.744 ± 0.015	$c = 64$	$0.739 \pm 0.010^\dagger$	37.7%	-0.005

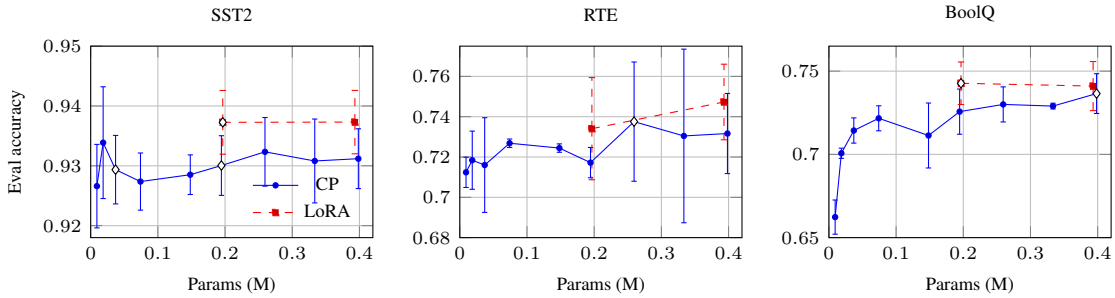

Figure 3. Low-budget zoom ($c \leq 43$, LoRA $r = 1, 2$). Error bars use all available seeds; diamond markers denote selected ten-seed cells. LoRA points correspond to rank choices, while CP points correspond to component counts.

Table 4. Selected ten-seed confirmation cells. These runs use the same protocol as the base grid and extend selected settings to seeds 0–9.

Task	Setting	Params	Opt. MB	Eval
SST2	LoRA $r = 1$	196,608	1.500	0.937 ± 0.005
SST2	CP $c = 4$	37,056	0.283	0.929 ± 0.006
SST2	CP $c = 21$	194,544	1.484	0.930 ± 0.005
RTE	LoRA $r = 6$	1,179,648	9.000	0.760 ± 0.015
RTE	CP $c = 28$	259,392	1.979	0.738 ± 0.030
RTE	CP $c = 64$	592,896	4.523	0.736 ± 0.013
BoolQ	LoRA $r = 1$	196,608	1.500	0.743 ± 0.013
BoolQ	CP $c = 43$	398,352	3.039	0.737 ± 0.012
BoolQ	CP $c = 64$	592,896	4.523	0.739 ± 0.010
BoolQ	CP $c = 85$	787,440	6.008	0.737 ± 0.012

are naturally indexed by physical axes such as space, time, and frequency, and tensor decompositions are already used for wireless signal processing and channel modeling (Chen et al., 2021). In such settings, tensor factors may align with meaningful axes rather than with an arbitrary reshape of a dense matrix.

Table 5. SST-2 mid-budget tensorization sensitivity, approximately matched trainable-parameter budgets, averaged over seeds $0, 1, 2$.

Split	Dev	Eval	Params	Opt. MB
row/col 32×64	0.925	0.933	787,440	6.008
row/col 16×128	0.929	0.932	790,704	6.033
row/col $16 \times 16 \times 8$	0.925	0.936	789,264	6.022

6.4. Limitations

The scope is narrow by design: one model scale (OPT-1.3B), three classification tasks, attention `q_proj`/`v_proj` targets only, and one main tensorization. The base grid uses three seeds, with selected key cells confirmed using ten seeds. Learning rates were chosen before the main grid, without a large per-method sweep. The tensorization sensitivity study was run on SST-2 only. Because CP has more tested capacity settings than LoRA, best-under-budget comparisons are descriptive and should be interpreted together with the raw curves and seed confirmations. The reported memory numbers count adapter optimizer state rather than total

training memory. We also did not grow c during training, tune tensorizations per layer, or allocate different component counts to different projections. Those choices require separate experiments.

7. Conclusion

We studied parameter-step size as a property of PEFT adapter families. Fixed-component CP tensor adapters provide a much finer budget grid than LoRA under the same OPT-1.3B target projections. Across SST-2, RTE, and BoolQ, the observed curves are task-dependent: SST-2 shows an early plateau, BoolQ rises and saturates slightly below LoRA, and RTE remains LoRA-favored. Finer parameter steps therefore help diagnose where the PEFT budget curve changes, but finer granularity alone does not guarantee a better accuracy–budget curve.

Acknowledgements

The work was supported by the Research Council of Finland under Grant 359848.

References

Albert, P., Zhang, F. Z., Saratchandran, H., van den Hengel, A., and Abbasnejad, E. Towards higher effective rank in parameter-efficient fine-tuning using Khatri–Rao product. arXiv:2508.00230, August 2025.

Bian, F., Zheng, J., Liu, Z., Luo, J., and Cai, J.-F. Finding low-rank matrix weights in DNNs via Riemannian optimization: RAdaGrad and RAdamW. In *The Thirty-Ninth Annual Conference on Neural Information Processing Systems*, 2025.

Chen, H., Ahmad, F., Vorobyov, S. A., and Porikli, F. Tensor decompositions in wireless communications and MIMO radar. *IEEE Journal of Selected Topics in Signal Processing*, 15(3):438–453, 2021. doi: 10.1109/JSTSP.2021.3061937.

Gu, Y., Zhou, W., Iacovides, G., and Mandic, D. TeRA: Vector-based random tensor network for high-rank adaptation of large language models. arXiv:2509.03234, September 2025.

Hounie, I., Kanatsoulis, C., Tandon, A., and Ribeiro, A. LoRTA: Low rank tensor adaptation of large language models. arXiv:2410.04060, October 2024.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Kolda, T. G. and Bader, B. W. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009. doi: 10.1137/07070111X.

Li, B., Zhang, Y., and Giannakis, G. B. Low-rank adaptation redux for large models. arXiv:2604.21905, April 2026.

Li, S., Luo, X., Tang, X., Wang, H., Chen, H., Luo, W., Li, Y., He, X., and Li, R. Beyond zero initialization: Investigating the impact of non-zero initialization on LoRA fine-tuning dynamics. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 35519–35535. PMLR, July 2025.

Liu, S.-Y., Wang, C.-Y., Yin, H., Molchanov, P., Wang, Y.-C. F., Cheng, K.-T., and Chen, M.-H. DoRA: Weight-decomposed low-rank adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32100–32121. PMLR, July 2024.

Marmoret, A., Bensaid, R., Lys, J., Gripon, V., and Leduc-Primeau, F. TensLoRA: Tensor alternatives for LoRA. arXiv:2509.19391, September 2025.

Sheng, H. N., Wang, Z.-Y., Yang, M., and So, H. C. AROMA: Autonomous rank-one matrix adaptation. arXiv:2504.05343, April 2025.

Veeramacheneni, L., Wolter, M., Kuehne, H., and Gall, J. Canonical rank adaptation: An efficient fine-tuning strategy for vision transformers. In *Forty-second International Conference on Machine Learning*, 2025. Poster.

Yang, M., Chen, J., Tao, J., Zhang, Y., Liu, J., Zhang, J., Ma, Q., Verma, H., Zhang, R., Zhou, M., King, I., and Ying, R. Low-rank adaptation for foundation models: A comprehensive review. arXiv:2501.00365, December 2024a.

Yang, Y., Zhen, K., Banijamali, E., Mouchtaris, A., and Zhang, Z. AdaZeta: Adaptive zeroth-order tensor-train adaption for memory-efficient large language models fine-tuning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 977–995, Miami, Florida, USA, November 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.56.

Yang, Y., Zhou, J., Wong, N., and Zhang, Z. LoRETTA: Low-rank economic tensor-train adaptation for ultra-low-parameter fine-tuning of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3161–3176, Mexico City, Mexico, June

2024c. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.174.

Zhang, F. and Pilanci, M. Riemannian preconditioned LoRA for fine-tuning foundation models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 59641–59669. PMLR, July 2024.

Zhang, Q., Chen, M., Bukharin, A., He, P., Cheng, Y., Chen, W., and Zhao, T. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023.