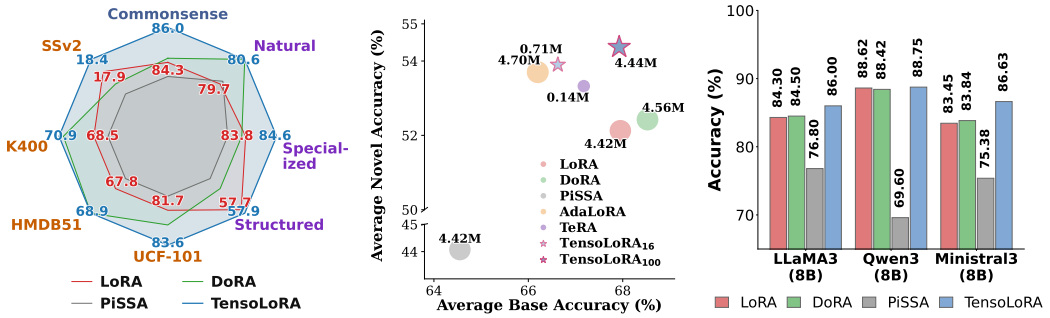
TENSOLORA: LEARNING HIGHER-ORDER REPRESENTATIONS OF WEIGHT UPDATES

Anonymous authors
Paper under double-blind review



(a) Performance across datasets (b) Performance per parameter (c) Performance across LLMs

Figure 1: Comparison of parameter-efficient fine-tuning methods from three perspectives: (a) dataset-wise performance, (b) the trade-off between trainable parameters and performance, and (c) commonsense reasoning performance across different LLM backbones.

ABSTRACT

Parameter-efficient fine-tuning (PEFT) adapts pretrained models under a limited trainable-parameter budget. Low-rank adaptation (LoRA) parameterizes each weight update with two low-rank factors. However, reducing the number of trainable parameters in LoRA requires a smaller rank, which lowers the matrix-rank bound and limits the expressive capacity of the weight update. At a fixed parameter budget, weight-update parameterization determines the matrix-rank bound and shapes how update magnitude is distributed across singular directions, as characterized by the singular-value (SV) spectrum. We therefore introduce Tensorized Low-Rank Adaptation (TensoLoRA), which learns higher-order representations of weight updates by jointly optimizing all tensor factors within a fixed parameter budget. Each tensor slice is expressed as a linear combination of shared dense planes, with the slice-wise coefficients parameterized by two line factors per plane. Factorizing these coefficients reduces the parameter cost per component relative to dense coefficient matrices, allowing more jointly learned plane-line components within the same budget while retaining dense trainable planes. TensoLoRA learns less concentrated SV spectra than LoRA and retains higher performance even when the final matrix rank is matched, as shown by rank-controlled analysis. Experiments on video base-to-novel generalization, commonsense reasoning, and image classification across diverse architectures show that TensoLoRA remains competitive with fewer trainable parameters and outperforms LoRA at matched parameter budgets.

1 INTRODUCTION

Recent advances in foundation models have enabled transferable representations across language, image, and video tasks. However, adapting foundation models to downstream tasks through full fine-tuning incurs substantial compute and memory costs and can compromise pretrained representations (Zhang et al., 2025; Kotha et al., 2024). Parameter-efficient fine-tuning (PEFT) reduces these costs and preserves pretrained representations by updating only a small set of task-specific parameters while freezing pretrained weights (Xu et al., 2026). LoRA (Hu et al., 2022) parameterizes each

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

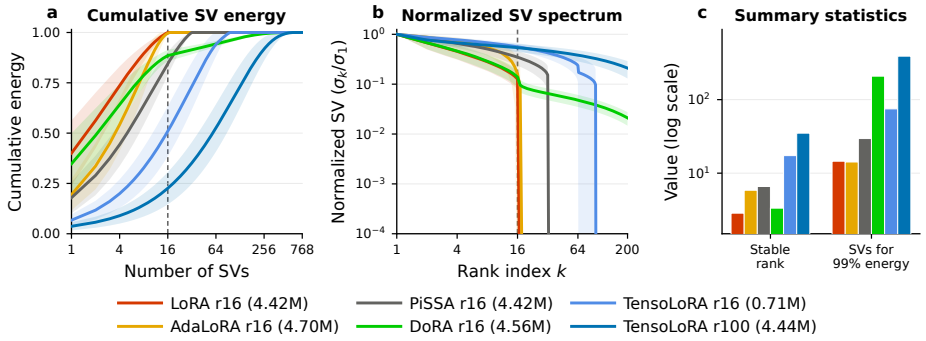


Figure 2: Singular-value (SV) spectra of weight updates learned on UCF-101. The spectra show how update magnitude is distributed across singular directions. (a) cumulative SV energy, (b) normalized SV spectra, and (c) stable rank and the number of SVs needed to capture 99% of the energy. Gray vertical dashed lines in (a) and (b) mark $k = 16$, the matrix-rank bound for LoRA with $r = 16$. Curves in (a) and (b) are normalized per projection and seed before equal-weight averaging over 144 attention and MLP projections and three seeds; shading shows the sample standard deviation across the 432 projection–seed observations. TensoLoRA exhibits less concentrated spectra than LoRA at both smaller ($r_{\mathrm{tl}} = 16$) and matched ($r_{\mathrm{tl}} = 100$) trainable-parameter counts.

weight update using two trainable low-rank factors, while AdaLoRA (Zhang et al., 2023) further allocates the rank budget across weight matrices based on their estimated importance. However, in LoRA, reducing the parameter budget requires a smaller rank and therefore lowers the matrix-rank bound of each weight update.

Recent works have explored PEFT methods that permit higher-rank weight updates than standard LoRA. For instance, MoRA (Jiang et al., 2024) uses a trainable square matrix, RandLoRA (Albert et al., 2025) constructs full-rank updates from fixed random low-rank matrices, and HiRA (Huang et al., 2025) uses a Hadamard product to produce high-rank updates. HiRA reports that LoRA updates have relatively low effective rank even with a LoRA rank of 512.

Motivation: This observation motivates examining the learned singular-value (SV) spectrum alongside the matrix-rank bound. **Problem Definition**: The matrix-rank bound limits the number of nonzero singular values, whereas the SV spectrum describes how update magnitude is distributed across singular directions. Prior work further suggests that LoRA's product parameterization influences the spectral structure of adapted weights (Shuttleworth et al., 2026). We therefore study how weight-update parameterization affects the learned SV spectrum and downstream performance while holding the number of trainable parameters fixed.

Tensorization offers a higher-order approach to weight-update parameterization by organizing adaptation parameters into tensors with multiple modes (Kolda & Bader, 2009). In PEFT, tensor factorizations have been used for parameter compression (Jie & Deng, 2023; Yang et al., 2024) and for high-rank weight updates. TeRA (Gu et al., 2026) tensorizes each weight update using frozen random tensor factors shared across layers and learns only layer-specific scaling vectors, enabling high-rank updates with few trainable parameters.

By contrast, we introduce Tensorized Low-Rank Adaptation (TensoLoRA), which learns higher-order representations of weight updates by jointly optimizing all tensor factors within a fixed parameter budget. Each tensor slice is expressed as a linear combination of shared dense planes, with the slice-wise coefficients factorized into two line vectors per plane. Factorizing these coefficients reduces the parameter cost per component relative to dense coefficient matrices, while retaining all entries of each plane as trainable parameters. The tensor shape and plane-line rank control the parameter cost of jointly learning the higher-order representation. Empirically, as shown in Figure 2, TensoLoRA learns less concentrated SV spectra than LoRA at both smaller and matched trainable-parameter counts.

Our main contributions are summarized as follows:

- We propose TensoLoRA, which learns higher-order representations of weight updates by jointly optimizing all tensor factors under a fixed parameter budget.

- We design a plane-line factorization that retains dense trainable planes and factorizes slice-wise coefficients. The tensor shape and plane-line rank control the parameter cost of learning the higher-order representation.
- We analyze SV spectra and perform rank-controlled interventions, showing that TensoLoRA outperforms LoRA at a matched final matrix rank.
- Across diverse datasets and backbones for commonsense reasoning, image classification, and video base-to-novel generalization, TensoLoRA remains competitive with fewer trainable parameters and outperforms LoRA at matched parameter counts (Figure 1).

2 Related Work

2.1 Matrix-Based Weight-Update Parameterizations

Recent efforts (Hu et al., 2022; Meng et al., 2024; Yun et al., 2025; Kopiczko et al., 2024; Jiang et al., 2024; Albert et al., 2025) have adapted pretrained models to downstream tasks by parameterizing task-specific weight updates in the original matrix space. For example, LoRA (Hu et al., 2022) and PiSSA (Meng et al., 2024) construct task-specific updates with low-rank matrix factors, while differing in how these factors are initialized. SoMA (Yun et al., 2025) adapts the minor singular components of the pretrained weight while keeping the complementary components fixed. To further reduce trainable parameters or support higher-rank weight updates, recent PEFT methods (Kopiczko et al., 2024; Jiang et al., 2024; Albert et al., 2025) employ fixed random factors or alternative matrix parameterizations. VeRA (Kopiczko et al., 2024) reduces trainable parameters by sharing fixed random low-rank matrices across layers, while MoRA (Jiang et al., 2024), RandLoRA (Albert et al., 2025), and HiRA (Huang et al., 2025) enable higher-rank weight updates through alternative matrix-space constructions. These approaches explore matrix-space constructions with different parameter costs and attainable matrix ranks. Our work examines how jointly learned higher-order weight-update representations affect the learned SV spectrum and downstream performance at a fixed trainable-parameter count.

2.2 Tensorized Parameter-Efficient Fine-Tuning

Tensor decompositions represent higher-order tensors through structured factorizations (Kolda & Bader, 2009). LoRETTA (Yang et al., 2024) applies tensor-train decomposition to trainable matrices in adapter and weight-reparameterization modules, reducing parameter and storage costs. FacT (Jie & Deng, 2023) jointly tensorizes weight increments across ViT layers to exploit redundancy within individual weights and across layers. TeRA (Gu et al., 2026) tensorizes each weight update using frozen random tensor factors shared across layers and learns only layer-specific scaling vectors, enabling high-rank updates with few trainable parameters. In comparison, TensoLoRA jointly learns all tensor factors of each higher-order weight-update representation under a fixed parameter budget. Motivated by the vector-matrix decomposition of TensoRF (Chen et al., 2022a), we express each tensor slice as a linear combination of shared dense planes, with the slice-wise coefficients factorized into two line vectors per plane. Factorizing only the slice-wise coefficients reduces the parameter cost without constraining the dense planes to rank-one outer products.

3 Preliminaries

3.1 Matrix Rank and Tensor Decomposition

For a matrix $M \in \mathbb{R}^{m\times n}$, $\mathrm{rank}(M)$ denotes its matrix rank, and a rank-r matrix can be represented as a sum of r rank-one matrices. To match the tensorization used in TensoLoRA, we consider an order-four tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K\times L}$. The CANDECOMP/PARAFAC (CP) decomposition of $\mathcal{X}$ is

$$\mathcal{X} = \sum_{s=1}^{R} a_s \otimes b_s \otimes u_s \otimes v_s, \tag{1}$$

where $a_s \in \mathbb{R}^I$, $b_s \in \mathbb{R}^J$, $u_s \in \mathbb{R}^K$, and $v_s \in \mathbb{R}^L$. The CP rank is the minimum number R of rank-one outer products required to represent $\mathcal{X}$ (Kolda & Bader, 2009).

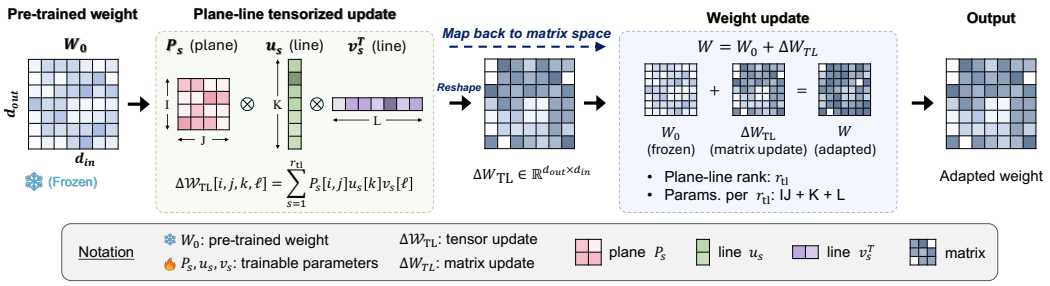


Figure 3: Overview of TensoLoRA. A plane-line tensor update is reshaped into matrix space and added to the frozen pretrained weight.

A matrix-valued weight update can be reshaped into an order-four tensor without changing the number of entries. Reshaping alone does not reduce the number of trainable parameters; the applied factorization determines both the parameter cost and the class of matrix updates that can be represented.

3.2 Problem Formulation

We consider adapting a pretrained model to a target task by learning additive updates to selected weight matrices. For a linear projection, let $x \in \mathbb{R}^{d_{\text{in}}}$ and $y \in \mathbb{R}^{d_{\text{out}}}$ denote the input and output feature vectors. The pretrained projection computes $y = W_0 x$, where $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ is the pretrained weight matrix. Let W denote the adapted weight matrix and $\mathcal{B}$ the maximum number of trainable parameters allocated to the update. We optimize the update while keeping W_0 fixed and satisfying the budget $\mathcal{B}$.

LoRA represents the update as a product of two trainable matrices:

$$W = W_0 + \Delta W_{\text{LoRA}}, \qquad \Delta W_{\text{LoRA}} = \frac{\alpha}{r} BA, \tag{2}$$

where $B \in \mathbb{R}^{d_{\text{out}} \times r}$ and $A \in \mathbb{R}^{r \times d_{\text{in}}}$ are trainable matrices, r is the LoRA rank, and α is a scaling hyperparameter. The update satisfies $\operatorname{rank}(\Delta W_{\text{LoRA}}) \le r$. The matrices A and B contain $r(d_{\text{out}} + d_{\text{in}})$ trainable parameters, so the budget requires

$$r \le \left\lfloor \frac{\mathcal{B}}{d_{\text{out}} + d_{\text{in}}} \right\rfloor. \tag{3}$$

Increasing r by one adds $d_{\text{out}} + d_{\text{in}}$ trainable parameters.

Tensorized PEFT represents adaptation parameters or weight updates with tensor factors θ subject to $N(\theta) \le \mathcal{B}$. We investigate how the choice of weight-update parameterization affects adaptation under a fixed trainable-parameter budget.

4 Methodology

4.1 Plane-Line Parameterization of Weight Updates

We parameterize each weight update with jointly learned dense planes and line factors. The planes are shared across tensor slices, with the line factors specifying the slice-wise coefficients. The construction is motivated by the matrix-vector factors used in TensoRF (Chen et al., 2022a).

Plane-Line Factorization. Choose positive integers (I, J, K, L) satisfying

$$IJKL = d_{\text{out}} d_{\text{in}}. \tag{4}$$

We arrange the update entries into KL slices of size $I \times J$, indexed by (k, ℓ). Each slice is represented as a linear combination of r_{tl} shared planes. For $s = 1, \ldots, r_{\text{tl}}$, let $P_s \in \mathbb{R}^{I \times J}$ denote a plane and $C_s \in \mathbb{R}^{K \times L}$ its coefficients across slices. Learning both P_s and C_s independently would require $IJ + KL$ trainable parameters.

TensoLoRA factorizes C_s into two line factors, $u_s \in \mathbb{R}^K$ and $v_s \in \mathbb{R}^L$:

$$C_s = u_s v_s^\top, \qquad C_s[k, \ell] = u_s[k] v_s[\ell]. \tag{5}$$

The plane-line rank r_{tl} is the number of factor triples (P_s, u_s, v_s). It is distinct from both the matrix rank of the reconstructed update ΔW_{TL} and the four-way CP rank. Factorizing C_s reduces its parameter count from KL to $K + L$ and constrains it to matrix rank at most one. Unlike four-way CP decomposition, which represents the first two modes with separate vectors, TensoLoRA learns all IJ entries of P_s directly. A dense plane uses IJ rather than $I + J$ parameters but avoids a rank-one constraint on P_s. The trainable variables are P_s, u_s, and v_s, so each unit increase in the plane-line rank adds $IJ + K + L$ trainable parameters.

The resulting update is an order-four tensor $\Delta \mathcal{W}_{\mathrm{TL}} \in \mathbb{R}^{I \times J \times K \times L}$:

$$\Delta \mathcal{W}_{\mathrm{TL}}[i, j, k, \ell] = \sum_{s=1}^{r_{\mathrm{tl}}} P_s[i, j] u_s[k] v_s[\ell]. \tag{6}$$

For fixed (k, ℓ), the corresponding slice is

$$\Delta \mathcal{W}_{\mathrm{TL}}[:, :, k, \ell] = \sum_{s=1}^{r_{\mathrm{tl}}} C_s[k, \ell] P_s = \sum_{s=1}^{r_{\mathrm{tl}}} \big(u_s[k] v_s[\ell]\big) P_s. \tag{7}$$

The tensor indices refer to weight entries, not spatial or temporal coordinates in the input.

Mapping Back to Matrix Space. We map the tensor update $\Delta \mathcal{W}_{\mathrm{TL}}$ back to matrix space by reshaping it to $d_{\mathrm{out}} \times d_{\mathrm{in}}$ and applying the scale α / r_{tl}:

$$\Delta W_{\mathrm{TL}} = \frac{\alpha}{r_{\mathrm{tl}}} \operatorname{reshape}\left(\Delta \mathcal{W}_{\mathrm{TL}}, d_{\mathrm{out}} \times d_{\mathrm{in}}\right). \tag{8}$$

We flatten the tensor in index order (i, j, k, ℓ), with ℓ varying fastest, and fill successive rows of the matrix without permuting the tensor axes. We set $\alpha = r_{\mathrm{tl}}$ for TensoLoRA, so the multiplier equals one across configurations.

4.2 Tensor Shape and Parameter Budget

The factorization in Sec. 4.1 makes the parameter requirement depend on both tensor shape and plane-line rank. We choose the factor dimensions and r_{tl} before training to satisfy the budget for each adapted projection.

Setting the Tensor Dimensions. For a positive common divisor d of d_{in} and d_{out}, we use

$$I = \frac{d_{\mathrm{in}}}{d}, \qquad J = \frac{d_{\mathrm{out}}}{d}, \qquad K = L = d. \tag{9}$$

The plane dimensions follow the projection widths, and the tensor size constraint gives $KL = d^2$. Among vector lengths with product d^2, the inequality

$$K + L \geq 2\sqrt{KL} = 2d \tag{10}$$

shows that $K = L = d$ minimizes the number of coefficient parameters for fixed plane dimensions.

Parameter Cost and Budget Constraint. Let $q(d)$ denote the trainable-parameter cost per unit of plane-line rank and N_{TL} the number of trainable adapter parameters in one projection. For the shape in equation 9, $q(d)$ and the budget condition are

$$\begin{aligned} q(d) &= \frac{d_{\mathrm{out}} d_{\mathrm{in}}}{d^2} + 2d, \\ N_{\mathrm{TL}} &= r_{\mathrm{tl}}(IJ + K + L) = r_{\mathrm{tl}} q(d) \leq \mathcal{B}. \end{aligned} \tag{11}$$

Each unit increase in the LoRA rank adds $d_{\mathrm{out}} + d_{\mathrm{in}}$ parameters, whereas each unit increase in the plane-line rank adds $q(d)$ parameters. A smaller $q(d)$ permits a larger r_{tl} at fixed parameter count, or fewer parameters at fixed r_{tl}. For fixed r_{tl}, increasing d reduces the plane size but increases both vector lengths.

4.3 Training and Inference

Initialization. For TensoLoRA, the adapted weight is $W = W_0 + \Delta W_{\mathrm{TL}}$. We initialize the factors so that the update is zero without modifying W_0. We assign nonzero values to P_s and u_s and set $v_s = 0$ for all s. Equation 6 then gives

$$\Delta W_{\mathrm{TL}}^{(0)} = 0, \qquad W^{(0)} = W_0. \tag{12}$$

The plane matrices use Kaiming uniform initialization. For $U = [u_1, \ldots, u_{r_{\mathrm{tl}}}] \in \mathbb{R}^{K \times r_{\mathrm{tl}}}$, we use orthonormal columns when $K \geq r_{\mathrm{tl}}$ and sample from $\mathcal{N}(0, 1/r_{\mathrm{tl}})$ otherwise. We initialize $V = [v_1, \ldots, v_{r_{\mathrm{tl}}}] \in \mathbb{R}^{L \times r_{\mathrm{tl}}}$ to zero.

Optimization and Inference. We keep W_0 and the bias frozen and optimize only the plane and line factors. With dropout $\mathcal{D}_p$ applied to the input of the adapter branch, the projection computes

$$y = W_0 x + \Delta W_{\mathrm{TL}} \mathcal{D}_p(x), \tag{13}$$

where p is the probability of dropping an input entry. Training reconstructs a dense matrix update and retains the dense projection operations. Storage for the trainable factors and their optimizer states scales with N_{TL}, but fewer trainable parameters do not by themselves guarantee lower training time or peak memory.

At inference time, we disable dropout and merge the learned update into the frozen weight:

$$W_{\mathrm{merged}} = W_0 + \Delta W_{\mathrm{TL}}. \tag{14}$$

The merged model uses the original linear projection without a separate adapter branch.

5 Experiments

In this section, we evaluate TensoLoRA across ● VIDEO, ● LANGUAGE, and ● IMAGE. We provide detailed experimental settings and additional results in the supplementary material (Section A). Across all tasks, we compare TensoLoRA with LoRA (Hu et al., 2022), AdaLoRA (Zhang et al., 2023), DoRA (Liu et al., 2024), and PiSSA (Meng et al., 2024). We further compare TensoLoRA with representative tensor-decomposition-based PEFT methods. TeRA (Gu et al., 2026) uses a Tucker-like tensor network for weight updates, while FacT (Jie & Deng, 2023) applies Tucker decomposition to tensorized ViT weight updates.

5.1 Video

Setup. We evaluate ● VIDEO on UCF-101 (Soomro et al., 2012), HMDB-51 (Kuehne et al., 2011), Kinetics-400 (Kay et al., 2017), and Something-Something V2 (SSv2) (Goyal et al., 2017). Following Chen et al. (2025), we adapt a pretrained ViCLIP-B/16 (Wang et al., 2024) using labeled samples from the base classes and evaluate it on both base and held-out novel classes. We report base and novel accuracy together with their harmonic mean (HM). The pretrained backbone remains frozen, and we optimize only the PEFT parameters.

Results. Table 1 compares TensoLoRA with existing PEFT methods under low and matched trainable-parameter counts. At $r_{\mathrm{tl}} = 16$, TensoLoRA uses 0.71M parameters, approximately 84% fewer than LoRA, while achieving higher HM on UCF-101, HMDB-51, and K-400 and higher novel-class accuracy than TeRA on all four datasets. At a matched parameter count, TensoLoRA with $r_{\mathrm{tl}} = 100$ uses 4.44M parameters and outperforms LoRA in HM on all four datasets. It also matches or achieves the highest novel-class accuracy across all four datasets, including a 3.2-point gain over LoRA on K-400 from 61.9 to 65.1. These results show that the weight-update parameterization affects both adaptation and novel-class generalization under a fixed trainable-parameter count.

5.2 Language

Setup. The ● LANGUAGE evaluation is conducted on commonsense reasoning with 170K training samples (Hu et al., 2022), with performance assessed on BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (HellaS.) (Zellers et al., 2019), WinoGrande

Table 1: Base-to-novel generalization with ViCLIP-B/16. "Zero-ViCLIP" denotes zero-shot inference with the pretrained backbone. Adapted methods use 12-view evaluation for Base and Novel (4 temporal clips $\times$ 3 spatial crops), with 8 frames per clip for HMDB-51 and 32 for the other datasets.

Method	Params. (M)	UCF-101			HMDB-51			K-400			SSv2		
		Base	Novel	HM	Base	Novel	HM	Base	Novel	HM	Base	Novel	HM
Zero-ViCLIP	149.6	80.7	71.7	75.9	66.0	55.6	60.4	67.2	62.1	64.6	8.7	8.3	8.5
+ LoRA	4.42	95.3±0.4	71.5±2.1	81.7±1.3	78.2±0.9	59.9±1.8	67.8±0.8	76.7±0.0	61.9±0.2	68.5±0.1	21.6±0.3	15.2±0.4	17.9±0.3
+ AdaptFormer	4.41	95.0±0.4	72.5±1.4	82.2±0.8	77.0±0.7	61.5±0.4	68.4±0.1	77.4±0.2	64.4±0.3	70.3±0.3	20.7±0.3	15.8±0.4	17.9±0.1
+ AdaLoRA	4.70	95.0±0.3	73.7±0.4	83.0±0.1	73.6±0.2	61.1±0.3	66.8±0.3	77.5±0.3	64.9±0.3	70.6±0.2	18.7±0.6	15.1±0.5	16.7±0.6
+ DoRA	4.56	**95.9**±0.1	72.4±0.9	82.5±0.5	**79.1**±1.4	61.1±1.1	**68.9**±1.0	77.4±0.1	61.7±0.8	68.6±0.5	**21.7**±0.1	14.5±0.7	17.4±0.5
+ PiSSA	4.42	93.9±0.4	60.3±1.0	73.4±0.7	74.5±0.7	56.1±0.2	64.0±0.3	71.0±0.4	45.9±0.5	55.7±0.2	18.8±0.3	14.0±0.6	16.0±0.4
+ TeRA	0.14	95.6±0.3	73.2±0.3	82.9±0.3	77.2±0.4	61.8±0.5	68.6±0.3	76.8±0.1	63.4±0.2	69.5±0.1	19.1±0.6	14.9±0.6	16.8±0.5
+ TensoLoRA ($r_{tl} = 16$)	0.71	95.2±0.2	73.9±0.2	83.2±0.1	75.2±0.5	62.1±0.8	68.0±0.3	77.4±0.3	64.5±0.3	70.4±0.3	18.7±0.7	15.1±0.2	16.7±0.4
+ TensoLoRA ($r_{tl} = 100$)	4.44	**95.9**±0.5	**74.0**±0.4	**83.6**±0.5	77.1±0.5	**62.2**±0.8	**68.9**±0.4	**77.8**±0.3	**65.1**±0.3	**70.9**±0.2	20.9±0.6	**16.3**±0.3	**18.4**±0.3

Table 2: Comparison of parameter-efficient adaptation methods on commonsense reasoning benchmarks using Llama 3-8B (Grattafiori et al., 2024). Ranks are configured based on LoRA with rank 32. Params. (M) denotes trainable parameters.

Method	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	Avg.	Params. (M)
LoRA	73.7 ±0.6	87.8 ±0.2	80.6 ±0.7	94.6 ±0.2	84.9 ±0.3	89.4 ±0.4	77.7 ±1.0	85.8 ±1.6	84.3 ±0.2	75.5
AdaLoRA	74.6 ±0.1	89.1 ±0.2	80.7 ±0.3	**96.1** ±0.1	86.1 ±0.5	91.3 ±0.2	**80.8** ±0.4	**87.5** ±1.0	85.8 ±0.2	75.5
DoRA	73.8 ±0.5	88.2 ±0.3	80.5 ±0.1	94.9 ±0.1	86.2 ±0.3	89.1 ±0.3	77.3 ±0.5	85.7 ±0.8	84.5 ±0.2	76.7
PiSSA	67.3 ±0.8	82.0 ±0.3	77.9 ±0.2	86.1 ±0.5	80.9 ±0.9	79.0 ±0.3	64.8 ±0.6	76.5 ±1.2	76.8 ±0.2	75.5
TeRA	65.8 ±2.5	83.4 ±1.0	69.5 ±8.9	86.7 ±2.7	75.4 ±1.3	81.8 ±7.1	69.1 ±6.6	71.7 ±11.0	75.4 ±4.5	0.3
TensoLoRA ($r_{\text{tl}} = 32$)	74.0 ±0.3	87.5 ±0.6	79.6 ±0.3	94.8 ±0.1	84.5 ±0.6	90.0 ±0.4	79.0 ±0.9	85.2 ±0.5	84.3 ±0.2	6.3
TensoLoRA ($r_{\text{tl}} = 384$)	**75.4** ±0.5	**89.2** ±0.6	**81.2** ±0.6	96.0 ±0.2	**86.4** ±0.4	**91.6** ±0.1	**80.8** ±0.5	87.3 ±1.6	**86.0** ±0.4	75.5

(WinoG.) (Sakaguchi et al., 2021), ARC-e and ARC-c (Clark et al., 2018), and OBQA (Mihaylov et al., 2018). We report the adaptation performance on language data using Llama3-8B (Grattafiori et al., 2024) as the primary backbone. Additional results and analyses for other LLM backbones, including Qwen3-8B (Team, 2025) and Ministral3-8B (Liu et al., 2026), are provided in Section B of the Supplementary Material.

Results. Table 2 compares TensoLoRA with existing LoRA-based methods on commonsense reasoning benchmarks. We evaluate LoRA at $r = 32$ and TensoLoRA at $r_{\text{tl}} = 384$, each using 75.5M trainable parameters. TensoLoRA achieves an average score of 86.0, a 1.7-point improvement over LoRA, with gains across all eight datasets. TensoLoRA is also evaluated with substantially fewer trainable parameters. At $r_{\text{tl}} = 32$, it uses 6.3M trainable parameters, 91.7% fewer than LoRA, while matching the average score of 84.3. These results show that TensoLoRA matches LoRA's average performance with fewer trainable parameters and improves performance at a matched parameter count.

5.3 IMAGE

Setup. We evaluate ● IMAGE on the Natural, Specialized, and Structured groups of VTAB-1K (Zhai et al., 2019). We set the rank for each task based on settings used in related works (Jie & Deng, 2023). We report two configurations: one using the same rank as LoRA, and another selected from $\{8, 32\}$ to achieve the best performance without exceeding the number of trainable parameters used by LoRA. We used a ViT-B model pretrained on ImageNet-21K as the backbone.

Results. Table 3 presents the image classification results on VTAB-1K. Following FacT (Jie & Deng, 2023), we set the LoRA rank to 8. For TensoLoRA, we report results with $r_{\text{tl}} = 8$ and the best-performing configuration $r_{\text{tl}} = 32$. At $r_{\text{tl}} = 8$, TensoLoRA achieves an overall mean of 73.9, outperforming LoRA by 0.1 points while using 82.4% fewer trainable parameters. Increasing r_{tl} to 32 raises the overall mean to 74.4, a 0.6-point improvement over LoRA, with 0.207M trainable parameters compared with 0.295M for LoRA.

5.4 ANALYSIS OF LEARNED WEIGHT UPDATES

Controlling for Matrix Rank. Table 4 tests whether final matrix rank explains TensoLoRA's advantage. The analyses in Table 4 and Table 5 cover all 144 attention and MLP projections across both encoders (72 per encoder). TensoLoRA with $r_{\text{tl}} = 100$ achieves an HM of 83.6, compared with 81.7 for LoRA with $r = 16$. After truncating all adapted TensoLoRA updates in both encoders to

Table 3: Results on VTAB-1K. We report task accuracy, group means (Mean), and group-wise averages (Overall Mean). The best and second-best results among PEFT methods are shown in bold and underlined, respectively. Params. (M) denotes trainable parameters.

Method	Natural								Specialized					Structured										
	CIFAR-100	Caltech101	DTD	Flowers102	Pets	SVHN	Sun397	Mean	Camelyon	EuroSAT	Resisc45	Retinopathy	Mean	Clevr-Count	Clevr-Dist	DMLab	KITTI-Dist	dSpr-Loc	dSpr-Ori	sNORB-Azim	sNORB-Elev	Mean	Overall Mean	Params. (M)
Full Fine-Tuning Methods																								
Full	68.9	87.7	64.3	97.2	86.9	87.4	38.8	75.9	79.7	95.7	84.2	73.9	83.4	56.3	58.6	41.7	65.5	57.5	46.7	25.7	29.1	47.6	68.9	85.8
Linear	64.4	85.0	63.2	97.0	86.3	36.6	51.0	69.1	78.5	87.5	68.5	74.0	77.1	34.3	30.6	33.2	55.4	12.5	20.0	9.6	19.2	26.9	57.7	0
BitFit	72.8	87.0	59.2	97.5	85.3	59.9	51.4	73.3	78.7	91.6	72.9	69.8	78.3	61.5	55.6	32.4	55.9	66.6	40.0	15.7	25.1	44.1	65.2	0.103
LoRA Methods																								
LoRA	70.8 ±2.3	96.3 ±0.9	64.7 ±0.2	**99.7** ±0.2	91.8 ±0.9	86.0 ±1.1	48.5 ±0.4	79.7 ±0.7	88.3 ±0.6	94.2 ±0.6	81.5 ±1.4	71.2 ±2.3	83.8 ±0.2	**82.7** ±1.7	61.0 ±2.2	48.3 ±3.3	80.3 ±1.7	86.3 ±2.3	50.7 ±2.3	18.5 ±0.7	33.5 ±2.9	57.7 ±0.5	73.8 ±0.6	0.295
AdaLoRA	71.3 ±0.9	96.0 ±0.7	**66.8** ±0.6	**99.7** ±0.2	92.5 ±0.4	87.0 ±0.7	51.0 ±0.8	80.6 ±0.3	**89.2** ±0.9	**94.3** ±0.6	**83.8** ±0.6	72.7 ±1.8	**85.0** ±0.5	81.8 ±0.6	60.5 ±2.3	49.7 ±2.1	77.2 ±2.1	**86.5** ±2.5	53.5 ±2.5	16.2 ±1.8	32.5 ±1.5	57.2 ±0.3	74.3 ±0.1	0.288
DoRA	70.7 ±1.3	96.5 ±0.7	65.2 ±1.2	98.8 ±0.5	90.8 ±0.5	87.5 ±1.5	49.0 ±0.8	79.8 ±0.6	88.3 ±1.3	93.0 ±0.4	80.5 ±1.8	73.0 ±0.0	83.7 ±0.8	**82.7** ±1.7	**62.0** ±0.8	**50.2** ±2.8	80.3 ±0.5	85.2 ±1.3	37.7 ±20.3	**21.2** ±0.6	33.0 ±2.6	56.5 ±2.7	73.3 ±1.4	0.323
PiSSA	68.3 ±2.7	**96.7** ±0.6	63.5 ±0.8	99.3 ±0.2	90.8 ±0.5	**88.5** ±0.4	46.5 ±1.5	79.1 ±0.7	87.2 ±0.9	93.0 ±0.7	78.2 ±1.3	73.5 ±1.5	83.0 ±0.9	77.7 ±1.6	57.8 ±1.4	46.2 ±2.1	76.2 ±5.1	85.7 ±0.9	51.2 ±1.6	18.0 ±1.8	**34.3** ±2.5	55.9 ±0.2	72.7 ±0.2	0.295
FacT-TK$_8$	71.0 ±2.7	95.0 ±0.8	65.0 ±0.8	99.5 ±0.4	94.0 ±0.7	87.7 ±0.9	**52.5** ±1.1	80.7 ±0.3	88.7 ±2.0	93.5 ±1.2	83.0 ±3.2	71.8 ±1.0	84.3 ±0.6	76.5 ±0.8	58.0 ±0.7	47.0 ±0.8	**81.8** ±1.0	81.8 ±2.3	**56.0** ±1.4	15.5 ±1.8	28.5 ±1.5	55.7 ±0.6	73.5 ±0.1	0.014
TeRA	63.8 ±1.4	94.5 ±0.0	65.7 ±0.9	**99.7** ±0.5	**94.2** ±0.2	76.5 ±1.1	49.3 ±0.5	77.7 ±0.1	86.8 ±0.5	91.7 ±1.2	80.2 ±0.6	74.2 ±0.2	83.2 ±0.3	64.0 ±0.7	52.8 ±1.6	43.5 ±2.3	76.3 ±2.4	66.2 ±2.4	47.0 ±0.8	14.7 ±0.5	24.0 ±2.7	48.6 ±1.3	69.8 ±0.5	0.010
TensoLoRA ($r_{tl} = 8$)	73.5 ±0.8	96.3 ±0.6	66.3 ±1.3	**99.7** ±0.5	91.8 ±0.6	87.3 ±1.3	52.0 ±0.4	**81.0** ±0.5	87.3 ±1.4	94.0 ±0.4	81.7 ±1.7	73.5 ±1.1	84.1 ±0.3	77.2 ±3.4	61.5 ±1.9	49.3 ±1.4	80.0 ±2.5	83.2 ±2.1	53.2 ±2.4	17.2 ±2.1	30.5 ±0.0	56.5 ±1.3	73.9 ±0.4	0.052
TensoLoRA ($r_{tl} = 32$)	**73.8** ±2.0	96.0 ±0.0	65.0 ±1.4	99.5 ±0.4	90.8 ±0.6	**88.5** ±0.4	50.2 ±1.0	80.6 ±0.5	88.7 ±0.2	94.0 ±0.7	80.8 ±3.3	**75.0** ±0.4	84.6 ±1.0	82.3 ±1.3	60.5 ±2.5	**50.2** ±2.4	80.3 ±0.6	85.7 ±0.5	54.7 ±1.7	20.2 ±0.6	29.3 ±0.9	**57.9** ±0.9	**74.4** ±0.8	0.207

Table 4: Rank-controlled analysis on UCF-101. TensoLoRA uses $r_{\text{tl}} = 100$. † denotes per-projection truncation to rank 16; "–" denotes no rank truncation.

Update	Matrix rank	Stable rank	SVs@99%	HM
LoRA ($r = 16$)	≤ 16	2.86$\pm$0.1	14.55$\pm$0.1	81.7$\pm$1.3
TensoLoRA	–	–	–	83.6$\pm$0.5
TensoLoRA†	16	**6.84**$\pm$0.1	**16.00**$\pm$0.00	83.0$\pm$0.4

Table 5: Rank-16 SV and singular-direction interventions between LoRA and TensoLoRA on UCF-101.

Update	HM
TensoLoRA†	83.0$\pm$0.4
LoRA directions + TensoLoRA SVs	80.1$\pm$0.0
TensoLoRA directions + LoRA SVs	82.1$\pm$0.2

matrix rank 16, HM decreases from 83.6 to 83.0 and remains 1.3 points above LoRA. The rank-16 TensoLoRA updates have a stable rank of 6.84, compared with 2.86 for LoRA, and require 16.00 versus 14.55 SVs to capture 99% of the energy. For Table 4, we average per-projection spectral statistics within each seed and report the mean and sample standard deviation of the three seed-level averages. Thus, after matching the final matrix rank, TensoLoRA retains higher performance and a less concentrated SV spectrum, indicating that the distribution of update magnitude across singular directions provides information beyond matrix rank alone.

Singular Values and Directions. Table 5 further examines rank-16 updates by interchanging SVs and singular directions between LoRA and TensoLoRA across all adapted projections in both encoders. For each adapted projection, we match the Frobenius norms of the intervened updates to that of the rank-16 TensoLoRA update. Replacing TensoLoRA directions with LoRA directions reduces HM from 83.0 to 80.1, while replacing the SV distribution with Frobenius-matched LoRA SVs reduces HM to 82.1. With both matrix rank and update norm controlled, the remaining performance differences highlight the roles of the SV distribution and singular directions in the learned update.

5.5 Analysis of the Plane-Line Parameterization

Factorization Structure. Figure 4 compares three factorization structures at $d = 128$, using 19 components for the all-lines variant and 16 for the other two structures. TensoLoRA $(IJ + K + L)$ factorizes only the slice-wise coefficients while retaining dense trainable planes, whereas the all-lines variant $(I + J + K + L)$ represents each component as an outer product of four line factors. At comparable parameter counts (0.71M vs. 0.74M), TensoLoRA improves base accuracy over the all-lines variant by 0.9–3.3 points across all four datasets, with comparable or higher novel accuracy and consistently higher HM. The dense-coefficient variant $(IJ + KL)$ uses 37.87M parameters and

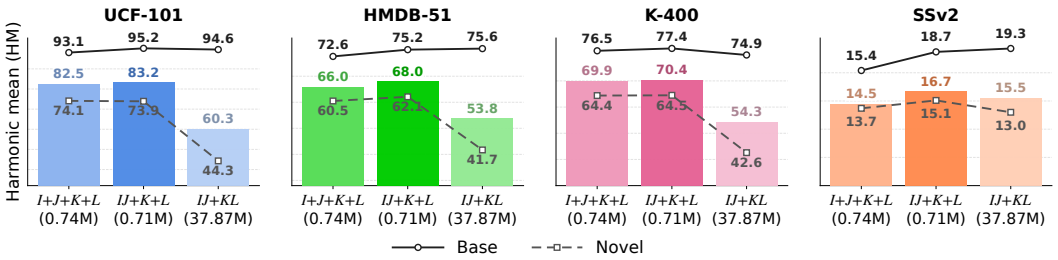


Figure 4: Factorization analysis at $d = 128$, with $r_{\text{tl}} = 19$ for $I + J + K + L$ and 16 otherwise. Bars show HM; labels report Base, Novel, and trainable parameters.

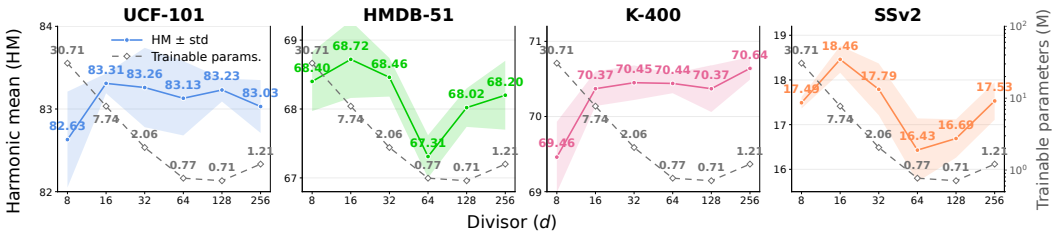


Figure 5: Analysis of the tensor divisor size across four video benchmarks.

yields lower novel accuracy and HM than TensoLoRA. These results support factorizing only the slice-wise coefficients while retaining dense, trainable planes.

Effect of the Tensor Divisor. Figure 5 examines the effect of the tensor divisor d at fixed $r_{\text{tl}} = 16$. The trainable-parameter count decreases from 30.71M at $d = 8$ to 0.71M at $d = 128$, then increases to 1.21M at $d = 256$. In contrast, HM varies non-monotonically with d and shows dataset-dependent trends. UCF-101, HMDB-51, and SSv2 achieve their highest HM at $d = 16$, whereas K-400 peaks at $d = 256$. These results show that tensor shape affects parameter cost and adaptation performance differently.

Table 6: Analysis of plane-line rank and parameter budget on base-to-novel generalization. The tensor shape is fixed while r_{tl} varies.

r_{tl}	Params. (M)	UCF-101	HMDB-51	K-400	SSv2
16	0.71	83.2±0.1	68.0±0.3	70.4±0.3	16.7±0.4
32	1.42	83.2±0.3	68.3±0.5	70.7±0.3	17.2±0.2
64	2.84	83.0±0.9	68.4±0.5	**70.9**±0.2	17.9±0.3
100	4.44	**83.6**±0.5	**68.9**±0.4	**70.9**±0.2	**18.4**±0.3
128	5.68	**83.6**±0.3	68.6±0.5	**70.9**±0.2	**18.4**±0.2
256	11.35	83.3±0.2	68.2±0.6	70.6±0.3	18.0±0.2

Plane-Line Rank and Parameter Budget. We fix the tensor shape and vary the plane-line rank r_{tl}, which directly changes the number of trainable parameters. As shown in Table 6, performance generally improves as r_{tl} increases from 16 to 100, then saturates. From $r_{\text{tl}} = 100$ to 128, HM remains unchanged on UCF-101, K-400, and SSv2 and decreases slightly on HMDB-51. At $r_{\text{tl}} = 256$, the parameter count reaches 11.35M while HM decreases on all four datasets relative to $r_{\text{tl}} = 100$. This non-monotonic trend shows that base-to-novel performance depends on the chosen plane-line rank rather than scaling monotonically with parameter count. Together with the matched-budget comparisons, these results show that adaptation depends on both parameter budget and weight-update parameterization.

6 Conclusion

We introduced Tensorized Low-Rank Adaptation (TensoLoRA), which tensorizes each weight update using a plane-line factorization. TensoLoRA learns less concentrated singular-value spectra than LoRA, and rank-controlled analysis shows that its performance advantage persists after matching the final matrix rank. Across commonsense reasoning, image classification, and video base-to-novel generalization, TensoLoRA remains competitive with substantially fewer trainable parameters and outperforms LoRA at matched parameter budgets.

AI USE STATEMENT

During the preparation of this work, the authors used ChatGPT (Singh et al., 2025) to improve readability and identify relevant literature. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

REPRODUCIBILITY STATEMENT

Details necessary for reproducing the results are described in the main paper and supplementary materials. We provide logs for all reported results in the supplementary materials and will publicly release the code upon publication.

REFERENCES

Paul Albert, Frederic Z. Zhang, Hemanth Saratchandran, Cristian Rodriguez-Opazo, Anton van den Hengel, and Ehsan Abbasnejad. RandLoRA: Full-rank parameter-efficient fine-tuning of large models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Sojung An, Junha Lee, Sujeong You, Nam Ik Cho, and Donghyun Kim. Loca: Spatially-aware low-rank convolutional adaptation of vision foundation models. In *European Conference on Computer Vision*, pp. 500–519. Springer, 2026.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020.

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensoRF: Tensorial radiance fields. In *Computer Vision – ECCV 2022*, pp. 333–350. Springer, 2022a.

Haoxing Chen, Zizheng Huang, Yan Hong, Yanshuo Wang, Zhongcai Lyu, Zhuoer Xu, Jun Lan, and Zhangxuan Gu. Efficient transfer learning for video-language foundation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 29129–29138, 2025.

Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. AdaptFormer: Adapting vision transformers for scalable visual recognition. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16664–16678, 2022b.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5842–5850, 2017. doi: 10.1109/ICCV.2017.622.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yuxuan Gu, Wuyang Zhou, Giorgos Iacovides, and Danilo Mandic. Tera: Vector-based random tensor network for high-rank adaptation of large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2314–2329, 2026.

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 *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Qiushi Huang, Tom Ko, Zhan Zhuang, Lilian Tang, and Yu Zhang. Hira: Parameter-efficient hadamard high-rank adaptation for large language models. In *International Conference on Learning Representations*, volume 2025, pp. 29614–29633, 2025.

Ting Jiang, Shaohan Huang, Shengyue Luo, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, Qi Zhang, Deqing Wang, and Fuzhen Zhuang. MoRA: High-rank updating for parameter-efficient fine-tuning. *arXiv preprint arXiv:2405.12130*, 2024.

Shibo Jie and Zhi-Hong Deng. FacT: Factor-tuning for lightweight adaptation on vision transformer. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(1):1060–1068, 2023. doi: 10.1609/aaai.v37i1.25187.

Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

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 *International Conference on Learning Representations*, 2024.

Suhas Kotha, Jacob Mitchell Springer, and Aditi Raghunathan. Understanding catastrophic forgetting in language models via implicit inference. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VrHiF2hsrm.

Hilde Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. HMDB: A large video database for human motion recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2556–2563, 2011. doi: 10.1109/ICCV.2011.6126543.

Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026.

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*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32100–32121. PMLR, 2024.

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*, volume 37, pp. 121038–121072, 2024.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2381–2391, 2018.

Hanoona Rasheed, Muhammad Uzair Khattak, Muhammad Maaz, Salman Khan, and Fahad Shahbaz Khan. Fine-tuned CLIP models are efficient video learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6545–6554, 2023.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, pp. 606–610. IEEE, 2007.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 4463–4473, 2019.

Reece Shuttleworth, Jacob Andreas, Antonio Torralba, and Pratyusha Sharma. Lora vs full fine-tuning: An illusion of equivalence. *Advances in Neural Information Processing Systems*, 38: 174627–174662, 2026.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human action classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Yi Wang, Yinan He, Yizhuo Li, Kunchang Li, Jiashuo Yu, Xin Ma, Xinhao Li, Guo Chen, Xinyuan Chen, Yaohui Wang, Ping Luo, Ziwei Liu, Yali Wang, Limin Wang, and Yu Qiao. InternVid: A large-scale video-text dataset for multimodal understanding and generation. In *International Conference on Learning Representations*, 2024.

Lingling Xu, Haoran Xie, S Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026.

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 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3161–3176, 2024.

Seokju Yun, Seunghye Chae, Dongheon Lee, and Youngmin Ro. SoMA: Singular value decomposed minor components adaptation for domain generalizable representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 25602–25612, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, Lucas Beyer, Olivier Bachem, Michael Tschannen, Marcin Michalski, Olivier Bousquet, Sylvain Gelly, and Neil Houlsby. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

Dan Zhang, Tao Feng, Lilong Xue, Yuandong Wang, Yuxiao Dong, and Jie Tang. Parameter-efficient fine-tuning for foundation models. *arXiv preprint arXiv:2501.13787*, 2025.

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 *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=lq62uWRJjiY.

A Experimental Settings

A.1 Video

We conduct experiments on four video action-recognition benchmarks, including Kinetics-400 (Kay et al., 2017), HMDB-51 (Kuehne et al., 2011), UCF-101 (Soomro et al., 2012), and Something-Something V2 (SSv2) (Goyal et al., 2017). Following the base-to-novel protocol of ViFi-CLIP (Rasheed et al., 2023), the action classes in each dataset are divided into disjoint base and novel subsets. The model is adapted using labeled training samples from the base classes only and evaluated on both the base and held-out novel classes. We employ the pretrained ViCLIP ViT-B/16 backbone (Wang et al., 2024) and use TensoLoRA for parameter-efficient downstream adaptation. The pretrained backbone weights remain frozen, and only the TensoLoRA parameters are optimized during training.

Following the official implementation of MSTA (Chen et al., 2025), we train the model for 11 epochs with a batch size of 32. We use multi-view inference for base-to-novel evaluation. We report the base-class accuracy, novel-class accuracy, and their harmonic mean (HM), which measures the balance between adaptation to seen classes and generalization to unseen classes. We compare TensoLoRA with AdaptFormer (Chen et al., 2022b), LoRA (Hu et al., 2022), PiSSA (Meng et al., 2024), and TeRA (Gu et al., 2026) under parameter-efficient adaptation settings. We apply parameter-efficient adaptation to both the text and video branches. The data augmentation strategy follows Chen et al. (2025), including ColorJitter, GrayScale, RandomResizedCrop, and horizontal flipping.

Table 7: Hyperparameters used for video recognition experiments.

Hyperparameter	**Value**
General Settings	
Optimizer	AdamW
Learning Rate	1×10^{-3}
Weight Decay	1×10^{-3}
Adam β_1	0.9
Adam β_2	0.98
Training Epochs	11
Warmup Epochs	5
Batch Size	32
LoRA Rank	16
LoRA Alpha Scale	1
Adapter Dropout (Vision)	0.1
Adapter Dropout (Text)	0.0
Seeds	$\{42, 1024, 2048\}$
Target Modules	attn.qkv, attn.proj, mlp.fc1, mlp.fc2
Data Augmentation	
ColorJitter	$P = 0.8$
GrayScale	$P = 0.2$
RandomResizedCrop	Applied
Horizontal Flip	$P = 0.5$
Method-specific Settings	
AdaLoRA Rank	$17 \rightarrow 16$
AdaptFormer Bottleneck	143
TensoLoRA Rank	$\{8, 16, 32, 64, 100, 128, 256\}$
TensoLoRA Divisor (Default)	128
TensoLoRA Divisor (Sweep)	$\{8, 16, 32, 64, 128, 256\}$

A.2 Language

Table 8 lists the hyperparameters used to tune the LLMs with LoRA-based methods. The rank is fixed to 32 for most methods to ensure a consistent comparison. AdaLoRA uses initial and target ranks of 48 and 24, respectively, selected through a search under a comparable parameter budget. We evaluate TensoLoRA at rank 32, as well as at higher-rank configurations (384, 375, and 366) matched to the parameter count of LoRA at rank 32.

Table 8: Hyperparameters used for language experiments.

Hyperparameter	Value
General Settings	
Optimizer	AdamW
Weight Decay	0
Base Model	{Llama-3-8B, Qwen3-8B, Ministral3-8B}
Learning Rate	3×10^{-4} (PiSSA on Qwen3-8B: 1×10^{-4})
Training Epochs	2
Warmup Steps	100
Batch Size	16
LoRA Dropout	0.05
LoRA Rank	32
LoRA Alpha Scale	1
Seeds	{42, 1024, 2048}
Target Modules	Llama3-8B: {q_proj, k_proj, v_proj, gate_proj, up_proj, down_proj } Qwen3-8B: {q_proj, k_proj, v_proj, gate_proj, up_proj, down_proj } Ministral3-8B: {q_proj, k_proj, v_proj, up_proj, down_proj}
Method-specific Settings	
AdaLoRA Rank	48→24
TensoLoRA Rank	32; {Llama3-8B: 384, Qwen3-8B: 375, Ministral3-8B: 366}
TensoLoRA Divisor	256

Table 9: Hyperparameters used for VTAB-1K experiments.

Hyperparameter	Value
General Settings	
Optimizer	AdamW
Base Model	ViT-B/16 pretrained on ImageNet-21K
Benchmark	VTAB-1K
Training Epochs	100
Warmup Epochs	10
Batch Size	64
Test Batch Size	512
Learning Rate Search	$\{1 \times 10^{-3}, 1 \times 10^{-2}\}$
Weight Decay Search	$\{1 \times 10^{-4}, 1 \times 10^{-3}\}$
LR Scheduler	Cosine decay
Minimum Learning Rate	1×10^{-5}
Warmup Learning Rate	1×10^{-6}
LoRA Dropout	0
LoRA Rank	8
LoRA Alpha Scale	1
Seeds	$\{42, 1024, 2048\}$
Target Modules	attn.qkv
Input Resolution	224×224
Drop Path Rate	0.1
Method-specific Settings	
AdaLoRA Rank	$10 \rightarrow 6$
TensoLoRA Rank	{8, 32}
TensoLoRA Divisor	256

A.3 IMAGE

We evaluate TensoLoRA on VTAB-1K (Zhai et al., 2019), which consists of 19 image-classification tasks grouped into Natural, Specialized, and Structured categories. We use a ViT-B/16 backbone pretrained on ImageNet-21K. Performance is measured by top-1 accuracy. We report the mean accuracy within each task group and the unweighted average of the three group means. We compare TensoLoRA with full fine-tuning, linear probing, and the LoRA methods LoRA (Hu et al., 2022), AdaLoRA (Zhang et al., 2023), DoRA (Liu et al., 2024), PiSSA (Meng et al., 2024), FacT (Jie & Deng, 2023), and TeRA (Gu et al., 2026). Following the experimental protocol of FacT (Jie & Deng, 2023), we perform a coarse grid search over two learning rates and two weight decay values and report the best-performing configuration. All models are trained for 100 epochs. For TeRA (Gu et al., 2026), we apply factorized adaptation to the attention QKV projection with `factor_dim_out`=48 and `factor_dim_in`=-1. The ViT-B/16 QKV weight has shape 2304×768, where $2304 = 48^2$, yielding a $48 \times 48 \times 768$ tensorized representation.

Table 10: Comparison of parameter-efficient adaptation methods on commonsense reasoning benchmarks. Ranks are configured based on LoRA with rank 32.

	Method	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	Avg.	Params. (M)
Llama3-8B	LoRA	73.7 ±0.6	87.8 ±0.2	80.6 ±0.7	94.6 ±0.2	84.9 ±0.3	89.4 ±0.4	77.7 ±1.0	85.8 ±1.6	84.3 ±0.2	75.5
	AdaLoRA	74.6 ±0.1	89.1 ±0.2	80.7 ±0.3	**96.1** ±0.1	86.1 ±0.5	91.3 ±0.2	**80.8** ±0.4	**87.5** ±1.0	85.8 ±0.2	75.5
	DoRA	73.8 ±0.5	88.2 ±0.3	80.5 ±0.1	94.9 ±0.1	86.2 ±0.3	89.1 ±0.3	77.3 ±0.5	85.7 ±0.8	84.5 ±0.2	76.7
	PiSSA	67.3 ±0.8	82.0 ±0.3	77.9 ±0.2	86.1 ±0.5	80.9 ±0.9	79.0 ±0.3	64.8 ±0.6	76.5 ±1.2	76.8 ±0.2	75.5
	TensoLoRA ($r_{\text{tl}} = 32$)	74.0 ±0.3	87.5 ±0.6	79.6 ±0.3	94.8 ±0.1	84.5 ±0.6	90.0 ±0.4	79.0 ±0.9	85.2 ±0.5	84.3 ±0.2	6.3
	TensoLoRA ($r_{\text{tl}} = 384$)	**75.4** ±0.5	**89.2** ±0.6	**81.2** ±0.6	96.0 ±0.2	**86.4** ±0.4	**91.6** ±0.1	**80.8** ±0.5	87.3 ±1.6	**86.0** ±0.4	75.5
Qwen3-8B	LoRA	**75.1** ±0.2	90.0 ±0.1	**81.5** ±0.4	95.5 ±0.1	**86.6** ±0.2	96.8 ±0.3	90.9 ±0.1	92.6 ±0.5	88.6 ±0.0	77.9
	AdaLoRA	73.7 ±0.2	90.9 ±0.3	80.9 ±0.7	93.6 ±2.4	85.1 ±0.4	97.4 ±0.1	92.3 ±0.2	92.1 ±0.3	88.2 ±0.4	77.8
	DoRA	74.8 ±0.3	90.2 ±0.3	81.4 ±0.6	95.4 ±0.1	86.3 ±0.4	96.7 ±0.4	90.6 ±0.2	92.1 ±1.0	88.4 ±0.2	79.1
	PiSSA	65.9 ±0.6	78.4 ±0.8	74.2 ±0.6	64.4 ±11.9	73.6 ±3.4	73.4 ±6.7	56.3 ±8.7	70.6 ±5.4	69.6 ±3.1	77.9
	TensoLoRA ($r_{\text{tl}} = 32$)	73.2 ±0.1	90.7 ±0.1	81.2 ±0.1	94.5 ±0.1	82.6 ±0.2	97.5 ±0.2	**92.7** ±0.2	90.7 ±0.3	87.9 ±0.0	6.6
	TensoLoRA ($r_{\text{tl}} = 375$)	74.6 ±0.3	**91.0** ±0.4	**81.5** ±0.2	**95.6** ±0.0	84.5 ±0.7	**97.6** ±0.2	92.5 ±0.4	**92.8** ±1.2	**88.8** ±0.3	77.8
Ministral3-8B	LoRA	70.9 ±0.6	86.7 ±0.3	80.7 ±1.0	93.4 ±0.1	85.6 ±0.2	88.2 ±0.3	75.9 ±1.1	86.0 ±1.2	83.5 ±0.2	59.0
	AdaLoRA	**74.5** ±0.2	89.0 ±0.2	**81.7** ±0.6	95.3 ±1.6	**88.2** ±1.1	**93.5** ±0.2	**84.2** ±0.1	86.7 ±0.6	**86.6** ±0.4	55.1
	DoRA	71.2 ±0.5	87.1 ±0.3	80.5 ±0.1	93.0 ±0.5	86.6 ±0.8	89.1 ±0.2	77.0 ±0.5	86.2 ±1.4	83.8 ±0.1	59.8
	PiSSA	67.3 ±0.2	80.4 ±0.6	77.8 ±0.5	78.2 ±0.3	79.4 ±0.4	78.8 ±1.0	63.5 ±0.7	77.7 ±0.4	75.4 ±0.3	59.0
	TensoLoRA ($r_{\text{tl}} = 32$)	72.4 ±0.6	88.1 ±0.6	78.3 ±4.3	91.3 ±6.0	86.3 ±1.4	92.5 ±0.1	82.6 ±1.1	85.9 ±0.6	84.7 ±1.2	5.2
	TensoLoRA ($r_{\text{tl}} = 366$)	74.2 ±0.1	**89.6** ±0.6	81.6 ±0.4	**95.8** ±0.1	87.6 ±0.2	92.9 ±0.3	83.0 ±0.7	**88.4** ±0.4	**86.6** ±0.0	59.0

Table 11: Analysis of prediction bias in binary classification on Ministral3-8B (BoolQ data). Bias denotes the deviation of the predicted true rate from the ground-truth label prior, and BA (balanced accuracy) denotes the mean recall across the two classes. All methods over-predict the majority class, with substantial differences in bias magnitude. The label-prior row reports the ground-truth proportion of True labels.

Method	Param. (M)	Predicted True Rate (%)	Bias	Recall_{T}	Recall_{F}	BA
Label prior	–	62.2	–	–	–	–
LoRA	59.0	70.3 ±2.3	+8.1	83.2 ±1.8	50.8 ±3.3	67.0 ±1.0
AdaLoRA	55.1	71.4 ±3.1	+9.2	**86.9** ±2.5	54.0 ±4.1	70.5 ±0.8
DoRA	59.8	70.4 ±2.4	+8.2	83.4 ±2.2	51.0 ±2.9	67.2 ±0.5
PiSSA	59.0	73.8 ±2.1	+11.6	83.1 ±1.7	41.5 ±2.9	62.3 ±0.6
TensoLoRA	59.0	**64.7** ±1.5	**+2.5**	81.2 ±1.3	**62.6** ±1.9	**71.9** ±0.3

B Generalization Across LLM Backbones on Language Tasks

Table 10 compares LoRA-based methods across different LLM backbones. We evaluate LoRA-based methods on three recent LLM backbones: Llama3-8B (Grattafiori et al., 2024), Qwen3-8B (Team, 2025), and Ministral3-8B (Liu et al., 2026). Llama3-8B follows the backbone used in Gu et al. (2026), while Qwen3-8B and Ministral3-8B extend the evaluation to more recent model families. All models follow the training settings in Section A.2, with method-specific rank and scaling settings. The learning rate is 3×10^{-4}, except for PiSSA on Qwen3-8B, which uses 1×10^{-4}.

The results show that TensoLoRA remains competitive with LoRA across the evaluated backbones. At $r_{\text{tl}} = 32$, TensoLoRA uses approximately 8–9% of the trainable parameters of LoRA. Its average score matches LoRA on Llama3-8B, is 0.7 points lower on Qwen3-8B, and is 1.2 points higher on Ministral3-8B. At comparable parameter counts, TensoLoRA improves the average score over LoRA by 1.7, 0.2, and 3.1 points on the three backbones, respectively. The largest improvement is observed on Ministral3-8B.

Table 11 examines class-wise prediction behavior on Ministral3-8B. True labels account for 62.2% of the BoolQ evaluation set, motivating an examination of majority-class overprediction and class-wise recall. At a matched parameter count, TensoLoRA reduces the excess True prediction rate from 8.1 to 2.5 percentage points relative to LoRA. False-class recall increases by 11.8 points, while True-class recall decreases by 2.0 points, yielding a 4.9-point improvement in balanced accuracy. These results show that TensoLoRA improves recognition of the minority False class while reducing majority-class overprediction and narrowing the gap between class-wise recalls.

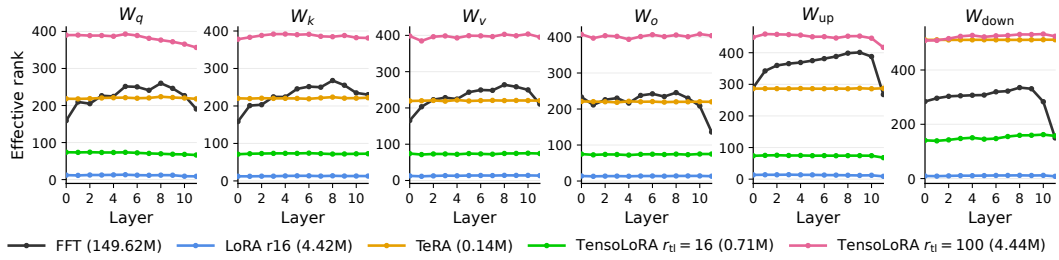


Figure 6: Effective rank across layers for FFT, LoRA, TeRA, and TensoLoRA.

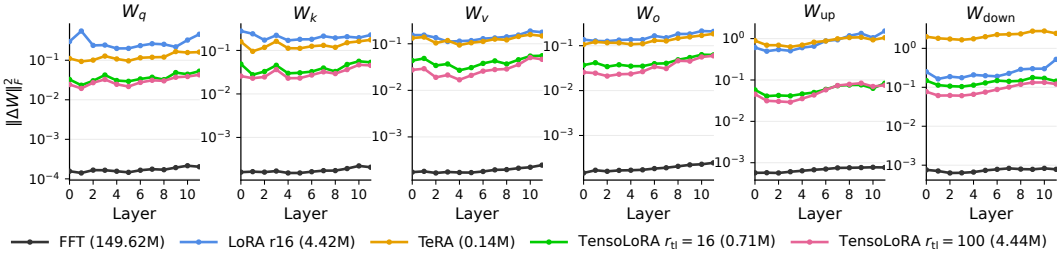


Figure 7: Sum of squared singular values across layers for FFT, LoRA, TeRA, and TensoLoRA.

C Layerwise Spectral Distribution and Update Magnitude

We report effective rank across ViCLIP-B/16 layers and projections after adaptation to UCF-101 in Figure 6. Matrix rank counts nonzero SVs, while a thresholded measure counts SVs above a prescribed cutoff. Effective rank summarizes the relative magnitudes of all SVs in the spectrum (Roy & Vetterli, 2007). Let $\sigma_1 \geq \cdots \geq \sigma_q \geq 0$ denote the singular values of a nonzero update ΔW, where $q = \min(d_{\text{out}}, d_{\text{in}})$. The effective rank is defined as

$$r_{\text{eff}}(\Delta W) = \exp\left(-\sum_{i=1}^{q} p_i \log p_i\right), \qquad p_i = \frac{\sigma_i}{\sum_{j=1}^{q} \sigma_j}. \tag{15}$$

With the convention $0 \log 0 = 0$, effective rank is scale invariant and satisfies $1 \leq r_{\text{eff}}(\Delta W) \leq \text{rank}(\Delta W)$. Larger values indicate a more even distribution of SV magnitude across singular directions. At matched trainable-parameter counts, TensoLoRA with $r_{\text{tl}} = 100$ (4.44M) yields higher effective rank than LoRA with $r = 16$ (4.42M) in every layer and projection. TensoLoRA with $r_{\text{tl}} = 16$ also yields higher effective rank than LoRA while using 0.71M parameters.

Figure 7 separates spectral spread from update scale by reporting the layerwise sum of squared singular values. For an update matrix ΔW, this quantity is defined as

$$\sum_{i=1}^{q} \sigma_i^2 = \|\Delta W\|_F^2. \tag{16}$$

This quantity equals the squared Frobenius norm and measures the overall scale of the update. Both TensoLoRA configurations have lower squared SV sums than LoRA and TeRA across all layers and projection matrices. At matched parameter counts, TensoLoRA with $r_{\text{tl}} = 100$ combines a higher effective rank with a lower update scale than LoRA with $r = 16$. These results show that TensoLoRA distributes update magnitude across more singular directions while maintaining a smaller update norm than LoRA.

D Structure and Evolution of Learned Updates

Figure 8 visualizes component-wise score maps for LoRA and TensoLoRA on Ministral3-8B under a comparable parameter budget. For LoRA, scores are based on the Frobenius norms of 256×256 input–output blocks in each component update $b_s a_s$. For TensoLoRA, scores are derived from plane coefficients scaled by the norms of the corresponding line factors, with coordinates indexing

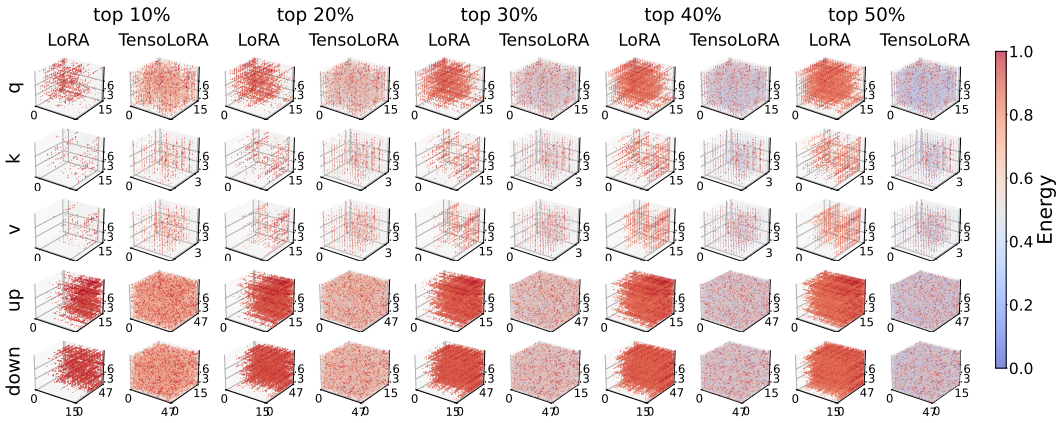


Figure 8: Component-wise score maps on commonsense reasoning (Ministral3-8B). Horizontal coordinates index matrix blocks for LoRA and tensor-plane entries for TensoLoRA; the vertical coordinate denotes the normalized component index. Each point represents a selected score-map entry. Colors indicate scores normalized by the 99.5th percentile within each adapter.

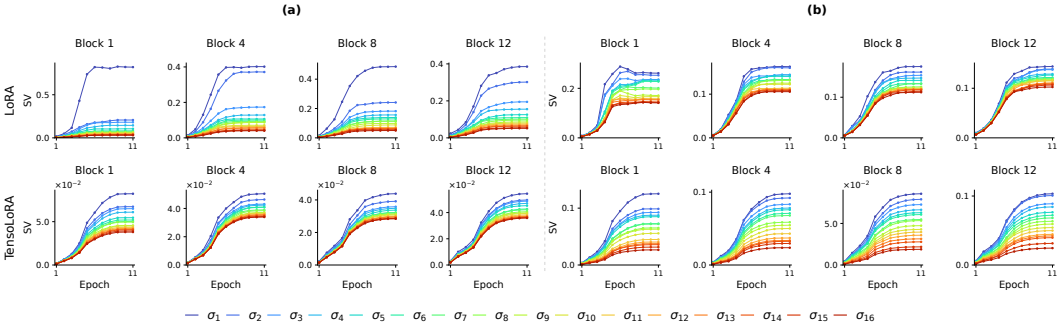


Figure 9: Evolution of update singular values on UCF-101. Rows compare LoRA ($r = 16$) and TensoLoRA ($r_{\text{tl}} = 16$, $d = 128$) at the key projections of ViCLIP-B/16 vision blocks 1, 4, 8, and 12. (a) Matrix SVD of ΔW; (b) SVD of the same update in the common $(IJ) \mid (KL)$ unfolding. Colors denote the 16 largest singular values at each of the 11 epochs (seed 1024). Values include adapter scaling without normalization; y-axis ranges are independent across subplots.

plane entries before the matrix reshape in Equation 8. Within each component, we display the top 10–50% of score-map entries. The maps provide a qualitative view of score distributions within each parameterization, complementing the matrix-level spectral analysis. Their coordinates are specific to each representation, and point counts depend on the number of learned components and the size of each score map.

Following LoCA (An et al., 2026), we track update singular values throughout training under the same training recipe. Figure 9(a) shows that LoRA develops a few dominant matrix singular values, whereas TensoLoRA exhibits growth across a broader set of singular modes. At the final epoch, σ_{16}/σ_1 ranges from 0.038–0.134 for LoRA and 0.462–0.673 for TensoLoRA across the displayed blocks, indicating a less concentrated leading spectrum. However, Figure 9(b) shows that TensoLoRA exhibits a more concentrated spectrum than LoRA in the common tensor unfolding. Together, these observations show that TensoLoRA combines a constrained tensor unfolding with a broader matrix-update spectrum, consistent with dense planes providing richer within-component representations. This spectral evidence complements the factorization ablation rather than independently establishing the cause of improved accuracy.

Table 12: Ablation of TensoLoRA across different target modules on video base-to-novel generalization with $r_{\text{tl}} = 16$ and divisor $d = 128$.

Target modules	Params. (M)	UCF-101			HMDB-51			K-400			SSv2		
		Base	Novel	HM	Base	Novel	HM	Base	Novel	HM	Base	Novel	HM
All (QKV+out+fc+proj)	0.71	95.2±0.2	**73.9**±0.2	**83.2**±0.1	75.2±0.5	**62.1**±0.8	**68.0**±0.3	**77.4**±0.3	**64.5**±0.3	**70.4**±0.3	**18.7**±0.7	**15.1**±0.2	**16.7**±0.4
Attn (QKV+out)	0.43	95.2±0.1	73.7±0.7	83.1±0.4	**75.3**±0.4	61.8±0.5	67.9±0.4	77.1±0.1	64.3±0.4	70.1±0.3	17.8±0.5	14.6±0.3	16.0±0.3
MLP (fc+proj)	0.28	**95.3**±0.2	73.0±0.6	82.7±0.4	75.0±0.2	61.9±0.7	67.8±0.5	76.6±0.1	64.0±0.4	69.8±0.3	17.3±0.4	14.2±0.2	15.6±0.2
QKV only	0.32	95.0±0.4	73.3±0.6	82.7±0.2	**75.3**±0.3	61.7±0.4	67.8±0.4	76.9±0.1	64.2±0.4	70.0±0.2	17.3±0.4	14.1±0.1	15.6±0.2
out_proj only	0.11	95.0±0.5	73.4±0.9	82.8±0.4	74.8±0.1	61.8±0.1	67.7±0.1	75.8±0.3	63.9±0.3	69.3±0.2	16.1±0.4	13.4±0.4	14.6±0.3
fc only	0.14	95.1±0.4	73.1±0.9	82.7±0.4	74.9±0.3	61.4±0.3	67.5±0.3	76.2±0.1	63.9±0.4	69.5±0.2	16.6±0.5	13.4±0.4	14.8±0.4
proj only	0.14	94.9±0.4	72.9±0.5	82.4±0.3	74.9±0.2	61.8±0.7	67.7±0.5	75.9±0.4	63.8±0.5	69.3±0.5	16.6±0.4	13.7±0.3	15.0±0.3

E Analysis on Target Module Selection

Table 12 compares TensoLoRA across different target modules. Applying TensoLoRA to all attention and MLP projections achieves the highest HM on all four datasets, with 83.2 on UCF-101, 68.0 on HMDB-51, 70.4 on K-400, and 16.7 on SSv2. Using only the attention projections gives the next highest HM on all four datasets and consistently outperforms the MLP-only setting. Within each block, adapting the full set of projections generally performs better than adapting a single projection. These results support adapting both attention and MLP projections in our main experiments.

F Computational Cost Comparison

This section presents a comparative analysis of GPU memory consumption and running time between the LoRA method and the proposed TensoLoRA. All training and evaluation runs use NVIDIA H100 GPUs for video experiments, NVIDIA B300 GPUs for language experiments, and NVIDIA RTX 4090 GPUs for image experiments. The computational costs in Table 13 are measured on the corresponding hardware. ● VIDEO uses HMDB-51, ● LANGUAGE uses commonsense reasoning benchmarks, and ● IMAGE uses the Pets dataset from VTAB-1K. TensoLoRA maintains training efficiency close to LoRA at the same rank. The increased cost at larger ranks reflects additional tensor reshaping.

Table 13: Computational efficiency comparison between LoRA and TensoLoRA. Training time is measured over 100 iterations after warm-up (10 iterations). All measurements use bfloat32 precision.

Task / Backbone	Metric	LoRA	TensoLoRA (Same Rank)	TensoLoRA (Matched Params.)
		Video		
ViCLIP-B/16	Params. (M)	4.42 ($r = 16$)	0.71 ($r_{\text{tl}} = 16$)	4.44 ($r_{\text{tl}} = 100$)
	Memory (GiB)	12.5	12.9	13.2
	Time / 100 Iter. (s)	35.3 ±0.9	36.0 ±0.8	50.2 ±0.8
		Language		
Llama3-8B	Params. (M)	75.5 ($r = 32$)	6.3 ($r_{\text{tl}} = 32$)	75.5 ($r_{\text{tl}} = 384$)
	Memory (GiB)	33.6	32.7	33.6
	Time / 100 Iter. (s)	294.1 ±0.0	297.8 ±0.0	362.3 ±0.0
		Image		
ViT-B/16	Params. (M)	0.295 ($r = 8$)	0.052 ($r_{\text{tl}} = 8$)	0.207 ($r_{\text{tl}} = 32$)
	Memory (GiB)	5.5	5.5	5.6
	Time / 100 Iter. (s)	24.3 ±0.4	24.5 ±0.3	25.8 ±0.0