TGRANK: TASK-GUIDED RANK LEARNING FOR LOW-RANK TENSOR REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-rank tensor representations model multidimensional feature tensors through factorized structures parameterized by continuous factorization variables and discrete ranks. While the factorization variables can be optimized end to end under task supervision, the ranks are typically prescribed before training and held fixed thereafter. Learning these ranks is nontrivial because rank decrease and rank increase operate on structures with different optimization histories: the former evaluates task-trained structure, whereas the latter must assess candidate structure whose task value has not yet been established. Motivated by this asymmetry, we propose TGRank, a task-guided framework that learns ranks during task optimization through adjacent rank changes. Specifically, TGRank employs direction-specific task evidence: rank decrease is guided by the task consequence of suppressing trained structure, whereas rank increase is guided by the task utility revealed after brief adaptation of candidate structure. Extensive experiments across super-resolution, image classification, and semantic segmentation demonstrate that TGRank learns nonuniform task-adaptive ranks and consistently outperforms matched fixed-rank counterparts.

1 INTRODUCTION

Low-rank tensor representations provide a compact factorized structure for multidimensional data and features, and have been extensively studied in machine learning and computer vision. Through their low-rank structure, these representations capture dependencies across tensor modes. They have been applied to multidimensional data recovery and inverse problems (Luo et al., 2022; 2024; Zeng et al., 2026), as well as to learned feature representations (Zhang et al., 2021; Dian et al., 2025).

In tensorized neural networks, low-rank tensor representations depend on continuous factorization variables and discrete ranks. The factorization variables are optimized end to end under downstream supervision, whereas the ranks determine the latent dimensionality of the factorized representation. For Tucker representations, mode-wise ranks determine the dimensions of latent subspaces (Tucker, 1966); for CANDECOMP/PARAFAC (CP), the rank determines the number of rank-one components (Kolda & Bader, 2009). Although these ranks control representational capacity, they are typically predefined before training and held fixed thereafter.

This raises the question of whether a rank specified before training remains suitable after task optimization. To investigate this, we perform controlled fixed-rank retraining across four Tucker representation sites by varying the mode-3 rank at each site while keeping the remaining ranks fixed. Figure 1 shows distinct rank–performance responses across sites: the saturation rank equals the initial rank at Site 0, is lower at Sites 1 and 3, and is higher at Site 2. These results suggest that, under the same retraining protocol, the rank required to reach performance saturation is not always the same as the pre-training rank and varies across representation sites, motivating the treatment of rank as a learnable structural variable during task optimization.

Existing rank learning methods adjust ranks according to predefined criteria, such as approximation quality, sparsity, parameter importance, or capacity allocation (Kodryan et al., 2023; Schotthöfer et al., 2022; Zhang et al., 2023; Liu et al., 2026). In contrast, rank adaptation in learned low-rank tensor representations in neural networks directly modifies task-trained representation structures. Such adaptation involves structures with different optimization histories across directions: structure

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.

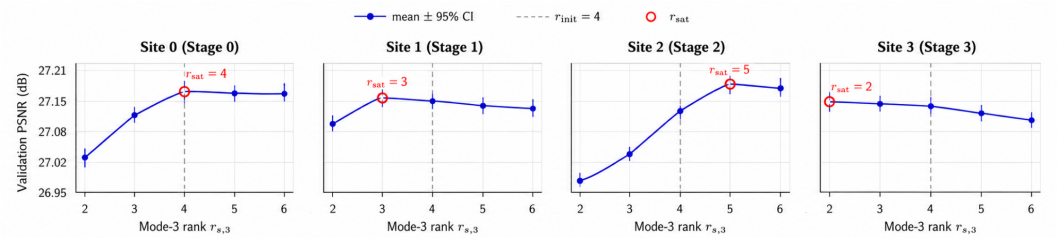


Figure 1: **Rank–performance responses across representation sites.** Validation PSNR under full retraining of fixed-rank variants while varying the Tucker mode-3 rank at one of four representation sites, each associated with a network stage, and holding all other ranks fixed. All sites share the same initialization $r_{\text{init}} = 4$, whereas the saturation rank r_{sat} differs across sites. Points denote the mean over five paired seeds, with error bars showing 95% confidence intervals. The saturation criterion is defined in Appendix A.4.

considered for removal has already been optimized jointly with the downstream task, whereas structure considered for introduction has not yet participated in task optimization. The available task evidence is therefore asymmetric across adaptation directions.

Motivated by this asymmetry, we propose **TGRank**, a task-guided framework for learning ranks of low-rank tensor representations during task optimization. TGRank uses direction-specific task evidence for rank decrease and rank increase, motivated by their different optimization histories. For rank decrease, the evidence comes from the task consequence of suppressing already-optimized structure; for rank increase, it comes from the task utility of candidate structure revealed after brief task-specific adaptation. This evidence guides adjacent rank changes during training. Extensive experiments across super-resolution, image classification, and semantic segmentation show that TGRank learns nonuniform task-adaptive ranks and outperforms fixed-rank counterparts, while further analyses show that TGRank can adjust ranks in both directions according to task demands rather than a predefined update direction.

In summary, our main contributions are as follows:

- We reveal an asymmetry in rank learning for learned low-rank tensor representations in neural networks: rank decrease removes already-optimized structure, whereas rank increase evaluates structure whose task value has not yet been established.
- We propose TGRank, a task-guided framework for rank learning that exploits this asymmetry through direction-specific task evidence. For rank decrease, it estimates the consequence of suppressing already-optimized structure; for rank increase, it evaluates candidate structure using task utility revealed after brief adaptation.
- We demonstrate the effectiveness of TGRank through extensive experiments across super-resolution, image classification, and semantic segmentation, showing that it learns nonuniform task-adaptive ranks and outperforms fixed-rank counterparts. Further analyses show that TGRank adapts ranks in both directions across different initializations.

2 Related Work

2.1 Low-Rank Tensor Representations

Low-rank tensor representations model multidimensional data and features through factorized structures, where ranks determine the dimensionality of the latent representation space. For example, Tucker decomposition represents a tensor using a core tensor and mode-wise factor matrices, with a tuple of mode-wise ranks specifying the latent dimensions of each mode; CP represents a tensor as a sum of rank-one components, with the rank determining their number; and tensor train organizes the representation through a sequence of tensor cores (Tucker, 1966; Kolda & Bader, 2009; Oseledets, 2011). Across these formulations, ranks therefore define the structural complexity and latent capacity of low-rank tensor representations, making their selection crucial for balancing compactness and expressive capability.

In neural networks with tensorized architectures, learned low-rank tensor representations have been explored for various tasks. CP-inspired rank-one tensors have been introduced as feature-domain priors for hyperspectral reconstruction (Zhang et al., 2021), LTRN learns low-rank spectral representations for spectral super-resolution (Dian et al., 2025), and F-INR uses functional tensor decomposition to construct implicit neural representations (Vemuri et al., 2026). Related low-rank tensor formulations have also been developed for multidimensional inverse problems using nonlinear transforms and latent generators (Luo et al., 2022; Zhou et al., 2025). These works primarily focus on learning the continuous factorization variables, while the ranks are typically specified in advance rather than learned jointly with the representation.

2.2 Rank Learning

Existing rank learning methods differ in the role assigned to rank and can be broadly divided into three categories. The first category focuses on tensor decomposition ranks, where rank serves as a structural parameter controlling representation complexity. Bayesian formulations infer model complexity through sparsity-inducing priors, including automatic CP-rank determination from incomplete observations (Zhao et al., 2015). Adaptive rank selection has also been studied for tensor-network decompositions; Sedighin et al. (2021) adjust Tensor Ring ranks according to approximation-error sensitivity. MARS instead learns relaxed binary masks over decomposition cores, allowing effective tensor ranks to emerge during optimization (Kodryan et al., 2023).

The second category considers ranks of factorized model parameters, where rank controls the parameter size and expressive capacity of low-rank weight representations. Dynamical low-rank training jointly evolves low-rank weight factors and their ranks during optimization (Schotthöfer et al., 2022). Geometry-aware Tucker training extends adaptive rank updates to Tucker-factorized weight tensors (Zangrando et al., 2024), while CoMERA combines rank-adaptive tensor optimization with tensorized model training for computational and memory efficiency (Yang et al., 2024).

The third category considers ranks of low-rank adapters for parameter-efficient fine-tuning, where rank controls the capacity of trainable parameter updates. LoRA parameterizes trainable weight updates with a fixed low rank while keeping pretrained weights frozen (Hu et al., 2022). Subsequent methods adapt the effective adapter rank or low-rank allocation budget during training. AdaLoRA reallocates the adaptation budget according to learned importance scores (Zhang et al., 2023); SoRA uses sparsity-inducing gates to adjust the effective adapter rank (Ding et al., 2023); GeoLoRA introduces adaptive allocation through dynamical low-rank geometry (Schotthöfer et al., 2025); and FlexLoRA supports both pruning and expansion of low-rank adaptation capacity (Liu et al., 2026).

These approaches adapt ranks associated with tensor decompositions, model parameters, or parameter updates, where rank primarily serves as a mechanism to control decomposition complexity, model capacity, or adaptation budget within an existing low-rank formulation. In contrast, TGRank studies the rank of task-trained tensor representations, where rank determines the latent capacity of the learned representation itself and is optimized jointly with downstream objectives. Therefore, rank learning in TGRank is formulated as a task-dependent structural learning problem.

3 Task-Guided Rank Learning

TGRank learns the ranks of task-trained tensor representations through local adjacent-rank changes guided by task evidence. We first formulate rank as a discrete structural variable coupled with ordinary task optimization, then define direction-specific evidence for rank decrease and rank increase, and finally specify the rank-update rule. Figure 2 summarizes the overall framework.

3.1 Problem Formulation

Let $F(\mathbf{x}; \boldsymbol{\Theta}, \mathbf{r})$ denote a tensorized neural network with low-rank tensor representations, mapping an input $\mathbf{x}$ to a task output with target $\mathbf{y}$, where $\boldsymbol{\Theta}$ collects the continuous trainable parameters and $\mathbf{r}$ contains bounded integer-valued rank variables. For a fixed $\mathbf{r} \in \mathcal{R}$, where $\mathcal{R}$ denotes the admissible rank space, the task objective is

$$\mathcal{L}(\boldsymbol{\Theta}; \mathbf{r}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\text{train}}} \left[\ell(F(\mathbf{x}; \boldsymbol{\Theta}, \mathbf{r}), \mathbf{y})\right], \tag{1}$$

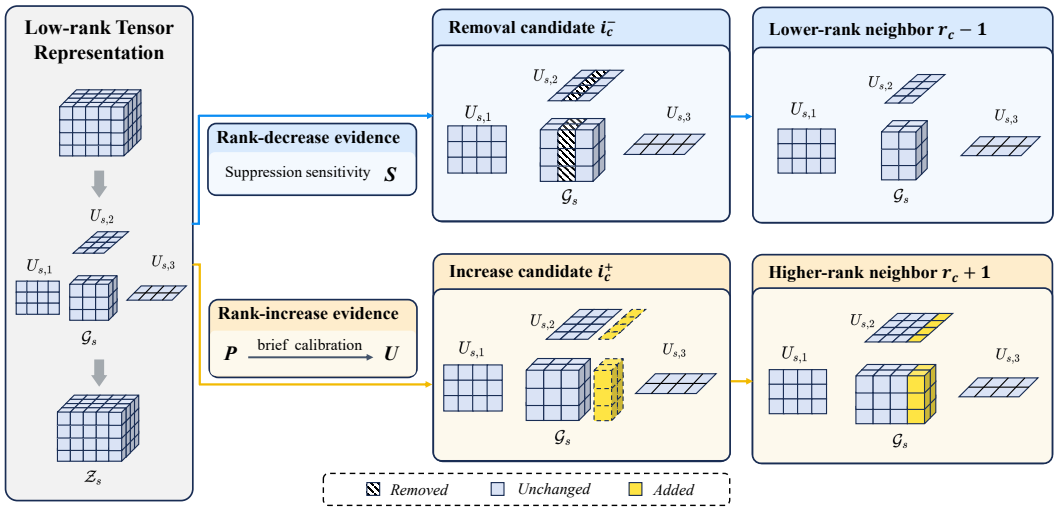


Figure 2: Overview of task-guided adjacent-rank learning in TGRank, illustrated with a Tucker representation. The lower- and higher-rank neighbors are evaluated using asymmetric task evidence from trained and candidate rank units, respectively. Rank decrease uses suppression sensitivity S, whereas rank increase uses P for candidate selection and U for evaluation after brief calibration. For Tucker, each one-step rank change jointly removes or introduces one mode-m factor direction and its matched core slice.

where $\mathcal{D}_{\text{train}}$ denotes the training distribution. Conventional training optimizes $\boldsymbol{\Theta}$ while treating $\mathbf{r}$ as an externally specified structural hyperparameter. TGRank instead treats $\mathbf{r}$ as a discrete structural variable and updates it during training according to task-derived evidence.

We consider elementary rank updates in which at most one rank coordinate changes by one unit. Let $\mathcal{C}$ denote the set of rank coordinates, and let $\mathbf{e}_c$ be the unit vector associated with coordinate $c \in \mathcal{C}$. At each rank evaluation, the candidate rank vector satisfies

$$\mathbf{r}' \in (\{\mathbf{r}\} \cup \{\mathbf{r} - \mathbf{e}_c,\ \mathbf{r} + \mathbf{e}_c :\ c \in \mathcal{C}\}) \cap \mathcal{R}. \tag{2}$$

The rank-update rule changes at most one coordinate of $\mathbf{r}$ by one unit or leaves $\mathbf{r}$ unchanged. Between rank evaluations, the continuous parameters $\boldsymbol{\Theta}$ are optimized under the current value of $\mathbf{r}$.

We instantiate this formulation for Tucker and CP representations. Let s index a representation site, i.e., a network location at which a factorized tensor representation is instantiated, and let M denote the tensor order. For Tucker,

$$\mathcal{Z}_s = \mathcal{G}_s \times_1 \mathbf{U}_{s,1} \cdots \times_m \mathbf{U}_{s,m} \cdots \times_M \mathbf{U}_{s,M}, \qquad \mathbf{r}_s = (r_{s,1}, \ldots, r_{s,m}, \ldots, r_{s,M}), \tag{3}$$

where $\mathcal{G}_s$ and $\{\mathbf{U}_{s,m}\}_{m=1}^{M}$ are generated from the feature at site s. Each Tucker rank coordinate is indexed by $c = (s, m)$, with $r_c = r_{s,m}$.

For CP,

$$\mathcal{Z}_s = \sum_{i=1}^{R_s} \mathbf{u}_{s,i}^{(1)} \circ \cdots \circ \mathbf{u}_{s,i}^{(m)} \circ \cdots \circ \mathbf{u}_{s,i}^{(M)}, \tag{4}$$

where $\mathbf{u}_{s,i}^{(m)}$ denotes the mode-m factor vector of the ith rank-one component at site s, and R_s denotes the number of rank-one components. Each CP site therefore has a single rank coordinate, indexed by $c = s$, with $r_c = R_s$.

For Tucker, $r_{s,m}$ specifies the latent dimension shared by the mode-m factor matrix and the corresponding mode of the core. An adjacent change in $r_{s,m}$ therefore removes or introduces one matched factor direction and core slice. For CP, the corresponding change removes or introduces one rank-one component. The adjacent-rank neighborhood in Eq. (2) therefore requires task evidence for assessing the lower- and higher-rank directions, together with an update rule for applying the supported change during training.

3.2 Task Evidence for Adjacent Ranks

Rank decrease. We use a *rank unit* to denote a mode-m factor direction with its matched core slice in Tucker, or a rank-one component in CP. Moving to the lower-rank neighbor removes one rank unit from the current representation. Since this unit has already been optimized during task training, the consequence of its removal can be estimated by temporarily suppressing its contribution.

Let $a_{c,i}$ be a scalar contribution coefficient for rank unit i. Setting $a_{c,i} = 0$ removes this contribution while leaving the remaining representation unchanged. Following the use of first-order Taylor approximations to estimate the loss consequence of removal (Molchanov et al., 2019), we write

$$\mathcal{L}(a_{c,i} = 0) - \mathcal{L}(a_{c,i}) \approx -a_{c,i} \frac{\partial \mathcal{L}}{\partial a_{c,i}}.$$

We therefore define the suppression sensitivity

$$S_{c,i} = \left| a_{c,i} \frac{\partial \mathcal{L}}{\partial a_{c,i}} \right|, \tag{5}$$

which estimates the local task consequence of removing rank unit i. TGRank accumulates this evidence during training rather than relying on a single-batch estimate. The removal candidate is the rank unit with the smallest accumulated suppression evidence, denoted by i_c^-.

Rank increase. Moving to the higher-rank neighbor introduces one rank unit into the current representation. Unlike existing rank units, the new unit has not been task-optimized, so its task value is not directly available. We initialize a candidate at zero contribution, which preserves the represented tensor. Let $g_{c,i}$ scale the contribution of candidate i, so that $\mathcal{Z}_{c,i}(g) = \mathcal{Z}_c + g_{c,i}\Delta\mathcal{Z}_{c,i}$.

At $g_{c,i} = 0$, the candidate contributes nothing and the current representation is unchanged. The multiplicative sensitivity used for rank decrease therefore collapses to

$$\left| g_{c,i} \frac{\partial \mathcal{L}}{\partial g_{c,i}} \bigg|_{g_{c,i}=0} \right| = 0. \tag{6}$$

This vanishes regardless of the value of the derivative $\partial\mathcal{L}/\partial g_{c,i}|_{g_{c,i}=0}$. The rank-decrease evidence cannot distinguish candidates at zero contribution, so we use the zero-state derivative magnitude to screen candidate directions before adaptation:

$$P_{c,i} = \left| \frac{\partial \mathcal{L}}{\partial g_{c,i}} \bigg|_{g_{c,i}=0} \right|. \tag{7}$$

The quantity P provides candidate evidence, whose accumulated value is used to identify the increase candidate i_c^+ for adaptation, but does not determine whether the rank should increase.

For the increase candidate i_c^+, we optimize its contribution for H steps on $\mathcal{B}_{\text{cal}}$ with shared model parameters fixed, and evaluate it on a disjoint batch $\mathcal{B}_{\text{eval}}$:

$$U_{c,i_c^+} = \mathcal{L}(\mathcal{B}_{\text{eval}}; g = 0) - \mathcal{L}(\mathcal{B}_{\text{eval}}; \tilde{g}), \tag{8}$$

where $\tilde{g}$ denotes the candidate coefficient after H calibration steps. The utility U measures the task benefit after brief adaptation and provides evidence for the higher-rank neighbor. The calibrated state is then discarded. If rank increase is selected, the candidate is introduced from zero contribution and optimized through ordinary task training. Complete update and evidence-accumulation details are provided in Appendix A.3.

3.3 Rank Update Rule

TGRank assesses the lower- and higher-rank neighbors, $r_c - 1$ and $r_c + 1$, using the task evidence above. The lower-rank neighbor is assessed through the current rank unit i_c^-, whereas the higher-rank neighbor is assessed through the candidate i_c^+ that is not yet part of the representation.

Because evidence magnitudes vary across rank coordinates, we normalize both directions using a coordinate-local reference. Let $\bar{S}$ and $\bar{U}$ denote accumulated suppression sensitivity and utility, respectively, and μ_c the mean suppression sensitivity of rank units at coordinate c. We define

$$q_c^- = \frac{\bar{S}_{c,i_c^-}}{\mu_c}, \qquad q_c^+ = \frac{\bar{U}_{c,i_c^+}}{\mu_c}. \tag{9}$$

Here, τ_- and τ_+ denote the support thresholds for rank decrease and rank increase, respectively. A rank decrease is supported when $q_c^- < \tau_-$, indicating weak task support for retaining the selected unit, whereas a rank increase is supported when $q_c^+ > \tau_+$, indicating sufficient utility from the calibrated candidate. A rank change becomes eligible only when the corresponding condition persists across repeated evaluations; otherwise, the current rank is retained.

If an eligible rank decrease is selected, the unit i_c^- is removed and $r_c \leftarrow r_c - 1$. If an eligible rank increase is selected, the candidate i_c^+ is introduced and $r_c \leftarrow r_c + 1$. For Tucker, the corresponding factor direction and matched core slice are removed or introduced jointly; for CP, the corresponding rank-one component is removed or introduced as a whole. At each evaluation, TGRank selects and applies at most one eligible adjacent rank change, consistent with the neighborhood defined in Eq. (2). Implementation details and the complete procedure are provided in Appendix A.3 and Algorithm A1, with sensitivity to the update thresholds and evidence normalization examined in Appendix D.5. For unique selections, the evidence-based unit choices are equivariant to matched permutations of Tucker latent directions and CP components; Appendix B provides the formal statement, proof, and empirical verification.

4 Experiments

4.1 Experimental Setup

We evaluate TGRank on three visual tasks: super-resolution, image classification, and semantic segmentation. Tucker representations are used across all three tasks, while CP is included as a structurally different low-rank tensor representation to examine whether the same rank-learning principle applies to other tensor factorization structures.

- **Super-resolution (SR).** We use MambaIRv2-Light (Guo et al., 2025) trained on DF2K (DIV2K and Flickr2K) (Agustsson & Timofte, 2017; Lim et al., 2017). The primary experiments use bicubic $\times 4$ super-resolution. Y-channel PSNR/SSIM is reported on Set5, Set14, B100, Urban100, and Manga109 (Bevilacqua et al., 2012; Zeyde et al., 2012; Martin et al., 2001; Huang et al., 2015; Matsui et al., 2017).
- **Image classification.** We use CIFAR-100 (Krizhevsky, 2009) with ResNet-18 (He et al., 2016) and evaluate both Tucker and CP representations. The official test set is used for final reporting.
- **Semantic segmentation.** We use ADE20K (Zhou et al., 2017) with SegFormer-B0 (Xie et al., 2021) and the official training/validation split. We additionally consider a low-rank initialization to examine rank increase under dense prediction.

Representation setting. Each representation is a third-order feature tensor with channel and two spatial modes, so $M = 3$ in our experiments. Representation sites are aligned with the stage structure of each backbone, yielding four sites for all three tasks considered here. At each site s, Tucker uses the mode-wise rank vector $\mathbf{r}_s = (r_{s,1}, r_{s,2}, r_{s,3})$, whereas CP uses a single rank R_s. For Tucker, site-wise rank tuples are separated by vertical bars; for CP, the corresponding scalar ranks are listed in the same site order. Task-specific representation and rank settings are provided in Appendix A.2.

Baselines. We compare TGRank with the corresponding backbone, matched fixed-rank Tucker/CP representations, and representative static and adaptive rank-selection mechanisms. For super-resolution, these include manual nonuniform ranks, HOSVD-based selection (Kolda & Bader, 2009), MARS-based selection (Kodryan et al., 2023), and FlexLoRA-based selection (Liu et al., 2026). Detailed baseline mappings are provided in Appendix A.6. Unless otherwise stated, results are reported over three paired seeds using matched data splits and training schedules.

4.2 Super-Resolution

We first evaluate TGRank on $\times 4$ super-resolution to examine whether task-guided rank learning can adapt representation capacity while preserving fine-grained reconstruction quality. Table 1 compares fixed, static, and adaptive rank-selection strategies under the same training protocol.

Table 1: Comparison of TGRank with fixed, static, and adaptive rank-selection strategies on super-resolution ($\times 4$). Results are three-seed mean PSNR/SSIM; standard deviations are reported in Tables C2 and C3. Final rank sums are comparable only within the same factorization family.

Method	Final Σr	Set5	Set14	B100	Urban100	Manga109
MambaIRv2-Light	–	32.51/0.8992	28.84/0.7878	27.75/0.7426	26.82/0.8079	31.24/0.9182
Fixed-rank representations						
Fixed CP $R = 8$	32	32.54/0.8994	28.87/0.7883	27.76/0.7430	26.88/0.8095	31.33/0.9189
Fixed Tucker $(8, 8, 4)$	80	32.53/0.8993	28.86/0.7881	27.76/0.7428	26.86/0.8091	31.30/0.9186
Fixed Tucker $(12, 12, 6)$	120	32.57/0.8997	28.90/0.7888	27.78/0.7433	26.92/0.8104	31.39/0.9192
Static rank selection						
Manual nonuniform	80	32.53/0.8994	28.87/0.7884	27.76/0.7430	26.87/0.8094	31.32/0.9188
HOSVD-based	78	32.51/0.8991	28.85/0.7879	27.75/0.7426	26.84/0.8087	31.27/0.9183
Adaptive rank-selection mechanisms						
MARS-based	79	32.58/0.8998	28.91/0.7890	27.78/0.7434	26.93/0.8107	31.40/0.9194
FlexLoRA-based	80	32.59/0.8999	28.92/0.7892	27.79/0.7437	26.95/0.8113	31.43/0.9197
TGRank–CP	17	**32.70**/0.9006	**29.01**/0.7906	**27.82**/0.7445	27.04/0.8139	31.62/0.9207
TGRank–Tucker	77	32.68/**0.9011**	**29.01/0.7908**	**27.82/0.7447**	**27.12/0.8151**	**31.64/0.9213**

Table 2: Rank learning on CIFAR-100. Results at epoch 200 are mean$\pm$std. over three paired seeds. Δ_{fixed} is measured relative to the matched fixed-rank representation.

Method	Top-1	Δ_{fixed}	Top-5	Δ_{fixed}	Σr: init.$\rightarrow$final
ResNet-18 (He et al., 2016)	75.70 ± 0.23	–	92.91 ± 0.20	–	–
Fixed CP	75.48 ± 0.20	–	92.84 ± 0.17	–	$22 \rightarrow 22$
TGRank–CP	$\mathbf{76.04 \pm 0.10}$	$\mathbf{+0.56}$	$\mathbf{92.98 \pm 0.10}$	$\mathbf{+0.14}$	$\mathbf{22 \rightarrow 13 \pm 3}$
Fixed Tucker	75.73 ± 0.16	–	92.84 ± 0.15	–	$48 \rightarrow 48$
TGRank–Tucker	$\mathbf{76.36 \pm 0.08}$	$\mathbf{+0.63}$	$\mathbf{93.05 \pm 0.08}$	$\mathbf{+0.21}$	$\mathbf{48 \rightarrow 36 \pm 3}$

TGRank–Tucker learns nonuniform ranks with a final rank sum of 77, lower than the standard fixed-rank setting (80) and the larger fixed-rank control (120), while outperforming all Tucker-based baselines. Its improvement over the larger-rank control indicates that the gain does not simply result from increased representation capacity. Comparisons with manually specified nonuniform ranks, HOSVD-based selection, and adaptive rank baselines further indicate that the benefit is not reproduced by nonuniform allocation or generic rank adaptation alone. TGRank–CP similarly reduces the total component count from 32 to 17 while maintaining strong reconstruction performance across all five benchmarks. Together, these results show that task-guided rank learning is effective across different low-rank tensor representations. The complete TGRank rank-learning procedure adds only 0.9% to end-to-end training time; detailed measurements are reported in Table E1. Corresponding qualitative comparisons on Urban100 and Manga109 are provided in Figure C1.

4.3 Image Classification

We next evaluate TGRank on CIFAR-100 to examine whether task-guided rank learning transfers to global recognition. Table 2 compares TGRank with matched fixed-rank Tucker and CP representations under the same training protocol. For Tucker, TGRank improves Top-1 accuracy by 0.63 points while reducing the rank sum from 48 to 36 ± 3. CP shows the same trend, with a 0.56-point Top-1 gain and a reduction from 22 to 13 ± 3 components. Both variants also improve Top-5 accuracy and outperform the ResNet-18 backbone, showing that TGRank learns more compact rank structures while improving performance.

4.4 Semantic Segmentation

We further evaluate TGRank on ADE20K with SegFormer-B0 to examine its behavior on semantic segmentation. Beyond task transfer, this setting directly tests whether the direction of rank change depends on the initial representation capacity.

Table 3: Rank learning on ADE20K with SegFormer-B0. Results and final rank sums are reported as mean±standard deviation over three seeds. Final ranks are shown for seed 20.

Method	mIoU	mAcc	Σr: init.→final	Final ranks
SegFormer-B0 (Xie et al., 2021)	37.40±0.10	47.53±0.11	–	–
Standard initialization				
+ Fixed Tucker	37.50±0.09	47.66±0.09	$80 \rightarrow 80$	$(8,8,4)\,\vert\,(8,8,4)\,\vert\,(8,8,4)\,\vert\,(8,8,4)$
+ **TGRank–Tucker**	**37.88±0.07**	**49.00±0.09**	$80 \rightarrow 66\pm1$	$(8,7,4)\,\vert\,(6,6,3)\,\vert\,(9,8,5)\,\vert\,(5,4,2)$
Low initialization				
+ Fixed Tucker	37.28±0.09	47.41±0.10	$40 \rightarrow 40$	$(4,4,2)\,\vert\,(4,4,2)\,\vert\,(4,4,2)\,\vert\,(4,4,2)$
+ TGRank–Tucker	37.61±0.09	47.79±0.10	$40 \rightarrow 58\pm1$	$(6,5,3)\,\vert\,(5,5,3)\,\vert\,(8,8,5)\,\vert\,(5,4,2)$

Table 4: Validation of task evidence for rank learning. Each evidence quantity is evaluated against an independently measured target matched to its function in TGRank.

Evidence	Function	Target	Metric	Baseline	Result
S	Removal consequence	Removal effect	Spearman ρ	Taylor: 0.55	**0.73**
P	Candidate selection	Best-gain candidate	Recall@1	Index: 42%	**67%**
U	Benefit evaluation	Beneficial increase	Precision	P-only: 71%	**83%**

Starting from the standard Tucker initialization, TGRank reduces the total rank from 80 to 66 ± 1, while improving mIoU from 37.50 ± 0.09 to 37.88 ± 0.07 and mAcc from 47.66 ± 0.09 to 49.00 ± 0.09, as shown in Table 3. The learned ranks vary across sites and modes, indicating nonuniform rank allocation rather than uniform reduction. With the low initialization, TGRank instead increases the total rank from 40 to 58 ± 1, while improving mIoU from 37.28 ± 0.09 to 37.61 ± 0.09 and mAcc from 47.41 ± 0.10 to 47.79 ± 0.10. Together, these results show that the same rank-learning procedure can decrease or increase rank according to task evidence rather than a prescribed direction. Qualitative comparisons are provided in Figure C2.

5 Analysis of Rank Learning

Beyond final task performance, we further analyze the validity of the proposed task evidence, the coordinate-wise dynamics of rank learning, and the effect of candidate calibration on rank-increase evidence.

5.1 Validation of Task Evidence for Rank Learning

We validate the task evidence used by TGRank through interventions at three predeclared checkpoints in the primary $\times 4$ super-resolution setting. For rank decrease, we suppress each S-selected candidate and measure the held-out PSNR drop. For rank increase, we optimize each feasible candidate under the same short-horizon budget to obtain reference gains, which test whether P identifies the highest-gain candidate and whether increases supported by U are beneficial. These measurements are used only for post-hoc analysis and do not affect rank learning. Full protocols are provided in Appendix A.5. Table 4 reports the Tucker results, with the corresponding CP analysis in Table A1. Accumulated suppression sensitivity S correlates more strongly with the measured removal effect than the instantaneous Taylor estimate ($\rho = 0.73$ vs. 0.55). For rank increase, P improves Top-1 recovery of the highest-gain candidate from 42% to 67%, while calibrated utility U improves beneficial-increase precision from 71% to 83% over P alone. These results support the asymmetric evidence design: S estimates removal consequence, whereas P and U separate candidate selection from benefit evaluation.

5.2 Rank Learning Dynamics

We next examine the dynamics of rank learning during task optimization. Figure 3 shows the Tucker rank trajectories across the four representation sites from the standard initialization. Although the sites begin from prescribed rank values, their trajectories diverge during training: some coordinates

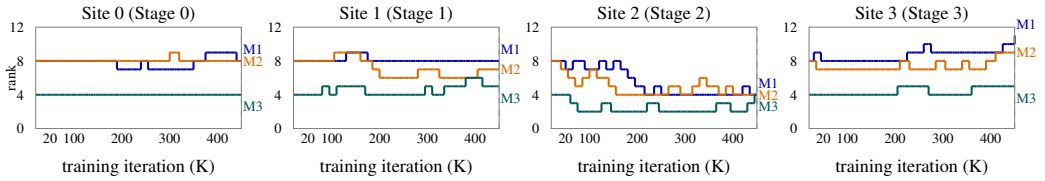


Figure 3: Training-time Tucker rank trajectories from the standard initialization in the primary $\times 4$ SR setting. Each panel shows the three mode-wise ranks at one representation site, with the corresponding backbone stage indicated in parentheses. Ranks change only at scheduled evaluations until 450k iterations, followed by fixed-rank refinement.

Table 5: Effect of candidate calibration horizon on Urban100 $\times 4$. $H = 0$ uses P without candidate adaptation, whereas $H > 0$ evaluates calibrated utility after H adaptation steps. Results are reported for seed 10, and rank-evaluation cost is measured per scheduled evaluation.

H	Evidence–gain ρ	Beneficial-increase precision	PSNR	Final Σr	Rank-eval. cost (s)
0	0.61	71%	26.86	79	18.4
1	0.71	82%	26.97	78	23.7
2	0.76	83%	27.12	77	29.6
4	0.77	90%	27.13	77	41.8

decrease, others increase, and several undergo intermediate changes before reaching their final values. The resulting rank structure is therefore neither a uniform reduction of the initial ranks nor a monotonic adjustment shared across representation sites.

Together, these trajectories show that rank learning is neither globally monotonic nor uniform across coordinates. Additional analyses separating the learned rank structure from the training path and examining initialization sensitivity are provided in Appendices D.2 and D.3, respectively.

5.3 Candidate Calibration Analysis

We analyze the predeclared candidate-calibration horizon H introduced in Section 3.2. Table 5 varies $H \in \{0, 1, 2, 4\}$ while keeping the rank-evaluation schedule and rank-update thresholds fixed. $H = 0$ removes candidate adaptation and uses P directly, whereas $H > 0$ evaluates calibrated utility after H adaptation steps.

Increasing the calibration horizon from $H = 0$ to $H = 2$ raises evidence–gain correlation from 0.61 to 0.76 and beneficial-increase precision from 71% to 83%, while PSNR changes from 26.86 to 27.12 dB. Extending the horizon to $H = 4$ yields only a small additional gain in correlation and PSNR, while increasing the rank-evaluation cost from 29.6 s to 41.8 s, although beneficial-increase precision further reaches 90%. These results support $H = 2$ as a favorable trade-off between evidence quality and evaluation cost.

6 Conclusion

We investigate rank learning for task-trained low-rank tensor representations, where rank evolves during task optimization rather than being prescribed before training. Specifically, TGRank updates rank through adjacent changes guided by direction-specific task evidence. Rank decrease evaluates the consequence of removing trained structure, while rank increase selects a zero-state candidate and evaluates its utility after brief adaptation. Across super-resolution, image classification, and semantic segmentation, TGRank learns nonuniform Tucker and CP ranks and outperforms matched fixed-rank baselines. Further analyses show rank changes in both directions and improved rank-increase evidence after candidate calibration, supporting learning rank jointly with continuous representations under the same task supervision.

AI USE STATEMENT

We used a large language model (ChatGPT) as a general-purpose assistant during manuscript preparation, primarily for language polishing, improving clarity and academic presentation, and providing suggestions on LaTeX formatting and code-level implementation checks. The language model was not used to generate experimental results or serve as a source of scientific evidence. All methodological decisions, mathematical formulations, experimental design, reported results, scientific claims, and conclusions were independently examined and verified by the authors, who take full responsibility for the content of the paper. The use of the language model did not constitute a substantive contribution warranting authorship.

REFERENCES

Eirikur Agustsson and Radu Timofte. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 126–135, 2017.

Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie-Line Alberi Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *Proceedings of the British Machine Vision Conference*, pp. 135.1–135.10. BMVA Press, 2012. doi: 10.5244/C.26.135.

Renwei Dian, Yuanye Liu, and Shutao Li. Spectral super-resolution via deep low-rank tensor representation. *IEEE Transactions on Neural Networks and Learning Systems*, 36(3):5140–5150, 2025. doi: 10.1109/TNNLS.2024.3359852.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. Sparse low-rank adaptation of pre-trained language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4133–4145. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.252.

Hang Guo, Jinmin Li, Tao Dai, Zhihao Ouyang, Xudong Ren, and Shu-Tao Xia. MambaIR: A simple baseline for image restoration with state-space model. In *European Conference on Computer Vision*, pp. 222–241. Springer, 2024.

Hang Guo, Yong Guo, Yaohua Zha, Yulun Zhang, Wenbo Li, Tao Dai, Shu-Tao Xia, and Yawei Li. MambaIRv2: Attentive state space restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 28124–28133, 2025.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

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.

Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5197–5206, 2015.

Maxim Kodryan, Dmitry Kropotov, and Dmitry Vetrov. MARS: Masked automatic ranks selection in tensor decompositions. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 3718–3732, 2023.

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

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Boyun Li, Haiyu Zhao, Wenxin Wang, Peng Hu, Yuanbiao Gou, and Xi Peng. MaIR: A locality- and continuity-preserving mamba for image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7491–7501, 2025.

Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. SwinIR: Image restoration using swin transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 1833–1844, 2021.

Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 136–144, 2017.

Muqing Liu, Chongjie Si, and Yuheng Jia. FlexLoRA: Entropy-guided flexible low-rank adaptation. In *International Conference on Learning Representations*, 2026.

Yisi Luo, Xi-Le Zhao, Deyu Meng, and Tai-Xiang Jiang. HLRTF: Hierarchical low-rank tensor factorization for inverse problems in multi-dimensional imaging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19303–19312, 2022.

Yisi Luo, Xi-Le Zhao, Zhemin Li, Michael K. Ng, and Deyu Meng. Low-rank tensor function representation for multi-dimensional data recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(5):3351–3369, 2024. doi: 10.1109/TPAMI.2023.3341688.

David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pp. 416–423, 2001.

Yusuke Matsui, Kota Ito, Yuji Aramaki, Azuma Fujimoto, Toru Ogawa, Toshihiko Yamasaki, and Kiyoharu Aizawa. Sketch-based manga retrieval using Manga109 dataset. *Multimedia Tools and Applications*, 76(20):21811–21838, 2017. doi: 10.1007/s11042-016-4020-z.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11264–11272, 2019.

Ivan V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011. doi: 10.1137/090752286.

Jinshan Pan, Long Sun, Lianhong Song, Jiangxin Dong, Jian Yang, Maocheng Zhao, and Jinhui Tang. Adaptive sparse self-attention for efficient image super-resolution and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(7):8283–8296, 2026. doi: 10.1109/TPAMI.2026.3670856.

Steffen Schotthöfer, Emanuele Zangrando, Jonas Kusch, Gianluca Ceruti, and Francesco Tudisco. Low-rank lottery tickets: Finding efficient low-rank neural networks via matrix differential equations. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Steffen Schotthöfer, Emanuele Zangrando, Gianluca Ceruti, Francesco Tudisco, and Jonas Kusch. GeoLoRA: Geometric integration for parameter efficient fine-tuning. In *International Conference on Learning Representations*, 2025.

Farnaz Sedighin, Andrzej Cichocki, and Anh-Huy Phan. Adaptive rank selection for tensor ring decomposition. *IEEE Journal of Selected Topics in Signal Processing*, 15(3):454–463, 2021. doi: 10.1109/JSTSP.2021.3051503.

Ledyard R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3): 279–311, 1966. doi: 10.1007/BF02289464.

Sai Karthikeya Vemuri, Tim Büchner, and Joachim Denzler. F-INR: Functional tensor decomposition for implicit neural representations. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 6557–6568, 2026.

Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, José M. Álvarez, and Ping Luo. SegFormer: Simple and efficient design for semantic segmentation with transformers. In *Advances in Neural Information Processing Systems*, volume 34, pp. 12077–12090, 2021.

Zi Yang, Ziyue Liu, Samridhi Choudhary, Xinfeng Xie, Cao Gao, Siegfried Kunzmann, and Zheng Zhang. CoMERA: Computing- and memory-efficient training via rank-adaptive tensor optimization. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Emanuele Zangrando, Steffen Schotthöfer, Gianluca Ceruti, Jonas Kusch, and Francesco Tudisco. Geometry-aware training of factorized layers in tensor tucker format. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Yiming Zeng, Xi-Le Zhao, Wei-Hao Wu, Teng-Yu Ji, and Chao Wang. Gaussian splatting-based low-rank tensor representation for multi-dimensional image recovery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19360–19369, 2026.

Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *Curves and Surfaces*, volume 6920 of *Lecture Notes in Computer Science*, pp. 711–730. Springer, 2012. doi: 10.1007/978-3-642-27413-8_47.

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

Shipeng Zhang, Lizhi Wang, Lei Zhang, and Hua Huang. Learning tensor low-rank prior for hyperspectral image reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12006–12015, 2021.

Qibin Zhao, Liqing Zhang, and Andrzej Cichocki. Bayesian CP factorization of incomplete tensors with automatic rank determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1751–1763, 2015. doi: 10.1109/TPAMI.2015.2392756.

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 633–641, 2017.

Ting-Wei Zhou, Xi-Le Zhao, Jian-Li Wang, Yi-Si Luo, Min Wang, Xiao-Xuan Bai, and Hong Yan. DTR: A unified deep tensor representation framework for multimedia data recovery. *IEEE Transactions on Multimedia*, 27:6347–6357, 2025. doi: 10.1109/TMM.2025.3581777.

Appendix

A Experimental Protocol and Implementation

A.1 Data Splits and Evaluation Separation

For super-resolution, we train on DF2K, consisting of the 800-image DIV2K training split and 2,650 Flickr2K images. The primary experiments use bicubic $\times 4$ degradation; the cross-scale study in Appendix D.6 additionally evaluates $\times 2$ and $\times 3$. The official 100-image DIV2K validation split is excluded from training and serves as the sole development set for rank-response diagnostics, saturation analysis, rank-update hyperparameters, and baseline selection. Set5, Set14, B100, Urban100, and Manga109 are reserved for final benchmark reporting using Y-channel PSNR/SSIM after removing the scale-dependent image border.

For CIFAR-100, protocol development uses a fixed stratified split of the official training set into 45,000 training and 5,000 validation images (50 validation images per class). The same split is used across methods and seeds for hyperparameter selection, diagnostic curves, and protocol decisions. Once these choices are frozen, final models are trained on the full 50,000-image training set for 200 epochs, and the official test set is evaluated only at the predeclared epoch-200 checkpoint. Seeds 10, 20, and 30 are paired across methods.

For ADE20K, the transfer study uses the official 20k/2k train/validation split and reports single-scale mIoU and mean class accuracy after 160k iterations. SegFormer-B0, fixed Tucker, and TGRank share each seed's initialization, data order, augmentation, optimizer, decoder, and checkpoint rule. The low-initialization stress test changes only the initial ranks. Evaluation uses the official validation split with single-scale inference.

Across all main comparisons, paired methods share the parameter initialization source, data permutation, crop sequence, augmentation, optimizer, scheduler, validation frequency, and checkpoint rule. Development choices are frozen before final training and benchmark evaluation.

A.2 Architectures, Rank Bounds, and Training Schedules

Shared representation structure. Across tasks, stage features are mapped to compact third-order tensors, represented by input-conditioned Tucker or CP factors, and reconstructed before being returned to the backbone. Task-specific interface dimensions and optimization settings are given below. All matched rank-selection comparisons use this same representation structure; the compared methods differ only in how the ranks are determined.

$\times 4$ **super-resolution.** MambaIRv2-Light is trained for 500k iterations using random 192×192 HR crops, horizontal flips, rotations, a global batch of 64 on two GPUs, Adam with $(\beta_1, \beta_2) = (0.9, 0.99)$, and an ℓ_1 objective. We use one fixed feature-to-tensor interface at each of the four selected backbone stages. The implementation applies a Haar transform, projects the low-frequency feature to 12 channels, partitions it into 16×16 patches, and generates input-conditioned Tucker factors and a core, or CP components. The reconstructed patches return through the inverse interface. Tucker initializes each stage at $(8, 8, 4)$ with bounds $(4, 4, 2)$–$(12, 12, 6)$, while CP initializes at rank 8 with bounds 4–12. The standard tuple serves as the common reference initialization, while these bounds define the admissible rank space for adjacent updates. Rank-update evaluations begin at iteration 20k, are performed every 5k iterations, and end at iteration 450k. The remaining iterations refine the continuous parameters under the learned ranks.

CIFAR-100 classification. ResNet-18 is trained for 200 epochs with batch size 128, SGD, momentum 0.9, weight decay 5×10^{-4}, and initial learning rate 0.1. Tucker initializes its four stage tuples at $(8, 8, 4) \,|\, (4, 4, 4) \,|\, (2, 2, 4) \,|\, (2, 2, 4)$. The corresponding lower and upper bounds are $(4, 4, 2) \,|\, (2, 2, 2) \,|\, (1, 1, 2) \,|\, (2, 2, 2)$ and $(12, 12, 6) \,|\, (6, 6, 6) \,|\, (3, 3, 6) \,|\, (2, 2, 6)$, respectively. CP initializes at $8|6|4|4$, with lower bounds $4|2|2|2$ and upper bounds $12|8|6|6$. Rank-update evaluations occupy the middle 86% of training and are performed every 1% of the training schedule. Epoch 200 is fixed before final training and is the only checkpoint evaluated on the official test set. Seeds 10, 20, and 30 are paired across methods.

ADE20K semantic segmentation. A Tucker representation is instantiated at each of the four MiT encoder stages of SegFormer-B0, with the decoder unchanged. The standard initialization is $(8, 8, 4)$ at every stage with bounds $(4, 4, 2)$–$(12, 12, 6)$; the low-initialization stress test uses $(4, 4, 2)$ at every stage. Training follows a 160k-iteration recipe with 512×512 random crops, AdamW, a polynomial learning-rate schedule, and cross-entropy loss. The first 10% of training precedes rank updates, the middle 80% permits rank-update evaluations every 2% of training, and the final 10% fixes the learned ranks for refinement. The backbone, fixed Tucker, and adaptive Tucker runs use three paired seeds and single-scale validation.

A.3 Rank-Update Implementation

This section details the implementation of the rank-update rule in Section 3.3, including candidate calibration, evidence accumulation, update eligibility, and selection among eligible changes.

Shared rank-update settings. Unless otherwise stated, we use $(\rho_S, \rho_P, \rho_U) = (0.95, 0.95, 0.80)$, $(\tau_-, \tau_+) = (0.35, 0.002)$, $K = 3$, $C = 2$, and $H = 2$ across tasks and decompositions. The raw-utility floor ϵ_U is set only to account for the task- and decomposition-specific loss scale: 10^{-7} for SR–Tucker and CIFAR–Tucker, 5×10^{-7} for the corresponding CP settings, and 2×10^{-6} for ADE20K–Tucker. Task-specific rank-evaluation schedules are given in Section A.2.

Candidate pool. For each rank coordinate c, candidate units are instantiated up to the admissible upper rank $r_c^{\max}$ at initialization. Units outside the current rank are retained as available candidates, with their structural parameters initialized using the same rule as the corresponding representation factors and their contribution coefficients fixed at zero. They therefore do not affect the current representation but define the candidate directions $\Delta Z_{c,i}$ used to evaluate $P_{c,i}$.

Candidate calibration. For the selected increase candidate i_c^+, let g denote its contribution coefficient, initialized as $g^{(0)} = 0$. This zero-contribution state leaves the current representation unchanged. With the shared model parameters fixed, one calibration step on $\mathcal{B}_{\text{cal}}$ is

$$g^{(h+1)} = g^{(h)} - \gamma \frac{\partial \mathcal{L}(\mathcal{B}_{\text{cal}}; g^{(h)})}{\partial g}. \tag{A1}$$

The zero-state derivative P is used to compare candidate directions before calibration. After H calibration steps, the selected candidate is evaluated on the disjoint batch $\mathcal{B}_{\text{eval}}$ to obtain U, and the temporary calibrated state is discarded. We use $H = 2$ unless otherwise stated.

Evidence accumulation. Suppression sensitivity S is computed throughout ordinary task training, whereas P and calibrated utility U are computed only at scheduled rank evaluations. We maintain separate exponential moving averages,

$$\bar{S} \leftarrow \rho_S \bar{S} + (1 - \rho_S) S, \qquad \bar{P} \leftarrow \rho_P \bar{P} + (1 - \rho_P) P, \qquad \bar{U} \leftarrow \rho_U \bar{U} + (1 - \rho_U) U. \tag{A2}$$

The $\bar{U}$ update is applied only to the candidate selected for calibration.

Candidate selection. For rank coordinate c, let $\mathcal{I}_c^{\text{cur}}$ denote the rank units currently included in the representation, and let $\mathcal{I}_c^{\text{cand}}$ denote the available units not yet included. The removal and increase candidates are selected directly from the corresponding evidence:

$$i_c^- = \arg\min_{i \in \mathcal{I}_c^{\text{cur}}} \bar{S}_{c,i}, \qquad i_c^+ = \arg\max_{i \in \mathcal{I}_c^{\text{cand}}} \bar{P}_{c,i}. \tag{A3}$$

Thus, i_c^- is the current rank unit with the weakest accumulated suppression evidence, whereas i_c^+ is the available candidate with the largest accumulated P value.

Normalized evidence. Evidence magnitudes can vary across rank coordinates. For coordinate c, we use the mean accumulated suppression sensitivity of the current rank units as the coordinate-local reference,

$$\mu_c = \frac{1}{|\mathcal{I}_c^{\text{cur}}|} \sum_{i \in \mathcal{I}_c^{\text{cur}}} \bar{S}_{c,i}. \tag{A4}$$

Algorithm A1 Task-guided adjacent-rank learning.

Require: Initial ranks $\mathbf{r}$; admissible rank bounds; scheduled rank evaluations
Initialize model parameters and evidence states $\bar{S}$, $\bar{P}$, $\bar{U}$.
for training iteration t **do**
  Update model parameters under the current $\mathbf{r}$ and accumulate suppression evidence $\bar{S}$.
  if t is a scheduled rank evaluation **then**
    Compute candidate evidence P for feasible rank increases and update $\bar{P}$.
    for each rank coordinate c **do**
      Select the removal candidate i_c^- from $\bar{S}$ and the increase candidate i_c^+ from $\bar{P}$.
      Temporarily calibrate i_c^+ when feasible, evaluate its utility U, update $\bar{U}$, and restore the pre-calibration state.
      Evaluate the eligibility of the lower- and higher-rank neighbors using q_c^- and q_c^+, with persistence across evaluations.
    end for
    Select at most one eligible adjacent rank change by the largest threshold-relative margin.
    Apply the selected change, if any: remove i_c^- for rank decrease or introduce i_c^+ from zero contribution for rank increase, and update $\mathbf{r}$.
  end if
end for
return Trained model and learned ranks $\mathbf{r}$.

The normalized evidence is then

$$q_c^- = \frac{\bar{S}_{c,i_c^-}}{\mu_c}, \qquad q_c^+ = \frac{\bar{U}_{c,i_c^+}}{\mu_c}. \tag{A5}$$

For numerical stability, coordinates with a degenerate normalization reference are excluded from the current evaluation. Let $r_c^{\min}$ and $r_c^{\max}$ denote the admissible lower and upper bounds of rank coordinate c. A rank decrease receives task support when

$$q_c^- < \tau_-,$$

whereas a rank increase receives task support when

$$q_c^+ > \tau_+.$$

Eligibility is additionally subject to the admissible rank bounds: $r_c > r_c^{\min}$ for decrease and $r_c < r_c^{\max}$ for increase. Rank increase further requires

$$U_{c,i_c^+}^{\mathrm{raw}} > \epsilon_U.$$

Here, $U_{c,i_c^+}^{\mathrm{raw}}$ denotes the calibrated utility measured at the current evaluation before temporal accumulation.

A decrease or increase becomes eligible only when its support condition, admissible-bound constraint, and the relevant increase requirements hold for K consecutive scheduled evaluations. For rank increase, the same candidate i_c^+ must remain selected throughout this persistence window.

Selecting one update. If multiple coordinate–direction pairs are eligible, TGRank compares their threshold-relative margins,

$$d_c^- = \frac{\tau_- - q_c^-}{\tau_-}, \qquad d_c^+ = \frac{q_c^+ - \tau_+}{\tau_+}. \tag{A6}$$

At most one adjacent rank change is applied at each scheduled evaluation, using the eligible change with the largest threshold-relative margin. The selected change updates $r_c \leftarrow r_c - 1$ for a decrease or $r_c \leftarrow r_c + 1$ for an increase. After an update, the affected coordinate is held fixed for the next C scheduled evaluations. Once rank adaptation ends, the learned ranks are fixed for the remaining task-training iterations.

A.4 Fixed-Rank Response Protocol

Each rank-response point is obtained by full fixed-rank retraining, with one rank coordinate varied and all remaining coordinates held at the reference values. Paired runs share the same initialization

and data order, and saturation is determined exclusively on DIV2K validation. Let $M_c(r)$ denote the validation metric obtained after retraining coordinate c at rank r, and let $M_c^\star = \max_r M_c(r)$. We define

$$r_c^{\mathrm{sat}} = \min\{r : U_{95}[M_c^\star - M_c(r)] \le \delta_{\mathrm{sat}}\}, \qquad \delta_{\mathrm{sat}} = 0.02\ \mathrm{dB}, \tag{A7}$$

$U_{95}[\cdot]$ denotes the upper endpoint of the paired 95% confidence interval on the performance deficit. A rank is therefore declared saturated only when its plausible deficit lies within the practical tolerance $\delta_{\mathrm{sat}} = 0.02$ dB. The mode-3 responses in Figure 1 use five paired seeds; the all-mode extension uses three paired seeds per coordinate. These fixed-rank experiments are offline diagnostics and do not affect TGRank training.

A.5 Intervention validation of rank evidence

We validate S, P, and U against independently measured task consequences at three predeclared checkpoints of the primary $\times 4$ super-resolution experiment. These interventions are used only for post-hoc analysis and do not affect rank learning.

For rank decrease, we suppress the unit selected by $\bar{S}$ and measure the resulting held-out PSNR drop. For rank increase, every feasible zero-contribution candidate is independently optimized under a fixed short-horizon budget and its held-out PSNR gain is recorded. These measurements define independent reference targets for evaluating removal consequence, candidate selection, and increase benefit.

We report Spearman correlation for removal consequence, Recall@1 for candidate selection, and precision for beneficial rank increases. The complete results are given in Table A1.

Table A1: Intervention validation of rank evidence on Tucker and CP. Spearman ρ measures association with the independent target. The primary metric is Recall@1 for increase-candidate selection and precision for increase-benefit evaluation.

Direction	Evidence function	Independent target	Signal	Tucker		CP	
				ρ	Primary metric	ρ	Primary metric
Decrease	Removal-consequence estimation	Held-out suppression effect	Coefficient magnitude	0.30	–	0.27	–
			Instantaneous Taylor	0.55	–	0.50	–
			EMA $\bar{S}$	**0.73**	–	**0.68**	–
Increase	Increase-candidate selection	Highest-gain increase candidate	Physical-index baseline	0.39	42%	0.35	38%
			P	**0.61**	**67%**	**0.56**	**63%**
Increase	Increase-benefit evaluation	Beneficial rank increase	P-based	0.58	71%	0.53	68%
			Calibrated utility U	**0.76**	**83%**	**0.71**	**79%**

A.6 Baseline Implementations

All super-resolution rank-selection baselines use the same Tucker representation, admissible rank bounds, training data, and evaluation protocol as TGRank; they differ only in how the ranks are determined. All development choices use DIV2K validation, while the reported SR benchmarks are reserved for final evaluation. Table A2 summarizes the matched Tucker instantiation of each selection principle. HOSVD uses a fixed 95% spectral-energy criterion, MARS applies relaxed masks to Tucker rank units, and FlexLoRA reallocates a fixed global rank budget across stage–mode coordinates. The labels "MARS-based" and "FlexLoRA-based" refer to these matched Tucker instantiations rather than direct applications of the original architectures.

Table A2: Matched rank-selection baselines on a common Tucker representation. Each variant preserves the defining rank-selection principle of the corresponding method while using the same representation structure and training protocol.

Method	Rank-selection principle	Tucker realization	Development protocol
Manual nonuniform	Predefined nonuniform ranks	Fixed stage–mode ranks	One DIV2K-val choice among predeclared equal-sum tuples
HOSVD-based	Spectral energy retention	Mode-unfolding spectrum per coordinate	Fixed 95% energy criterion; no search
MARS-based	Relaxed rank masking	Mask per Tucker rank unit	Four λ values; selected on DIV2K-val PSNR
FlexLoRA-based	Bidirectional budget reallocation	Stage–mode coordinates as allocation sites	Two score variants; fixed rank-sum budget $\sum r = 80$
TGRank	Task-evidence-guided adjacent rank updates	Stage–mode rank coordinates	Shared update thresholds; no benchmark-specific tuning

B PERMUTATION CONSISTENCY

Rank-unit indices are not intrinsic to Tucker or CP representations. For Tucker, jointly permuting a factor direction and its matched core slice leaves the represented tensor unchanged, while for CP the same holds for a common permutation of rank-one components. Since TGRank selects individual units when evaluating adjacent ranks, we examine whether these structural choices are consistent under such matched relabelings.

B.1 FORMAL PROPERTY

Proposition 1 (Permutation consistency)**.** *Consider two equivalent Tucker or CP representations related by a matched permutation σ of the latent units for rank coordinate c. The evidence-based unit selections for the lower- and higher-rank neighbors transform equivariantly under σ. When the selections are unique,*

$$i_c^{-\prime} = \sigma(i_c^-), \qquad i_c^{+\prime} = \sigma(i_c^+), \tag{B1}$$

and the corresponding adjacent-rank representations are equivalent under the same relabeling.

Matched relabeling. For Tucker coordinate $c = (s, m)$, let $\mathbf{\Pi}_\sigma$ denote the permutation matrix associated with σ. Relabeling the mode-m factor and its matched core mode as

$$\mathbf{U}'_{s,m} = \mathbf{U}_{s,m}\mathbf{\Pi}_\sigma, \qquad \mathcal{G}'_s = \mathcal{G}_s \times_m \mathbf{\Pi}_\sigma^\top \tag{B2}$$

preserves their product because $\mathbf{\Pi}_\sigma\mathbf{\Pi}_\sigma^\top = \mathbf{I}$:

$$\mathcal{G}'_s \times_m \mathbf{U}'_{s,m} = \mathcal{G}_s \times_m \mathbf{U}_{s,m}. \tag{B3}$$

Thus, the permutation changes only the labels of the matched Tucker latent units. For CP, applying the same permutation to the component columns of every factor matrix only reorders the rank-one terms in

$$\mathcal{Z}_s = \sum_{i=1}^{R_s} \mathbf{u}_{s,i}^{(1)} \circ \cdots \circ \mathbf{u}_{s,i}^{(M)}, \tag{B4}$$

and therefore also leaves the represented tensor unchanged.

Proof. Under a matched relabeling, latent unit i is reindexed as $\sigma(i)$ without changing the represented model or task loss. The corresponding unit-wise evidence therefore satisfies

$$S'_{c,\sigma(i)} = S_{c,i}, \qquad P'_{c,\sigma(i)} = P_{c,i}, \tag{B5}$$

and componentwise accumulation gives

$$\bar{S}'_{c,\sigma(i)} = \bar{S}_{c,i}, \qquad \bar{P}'_{c,\sigma(i)} = \bar{P}_{c,i}. \tag{B6}$$

Since the current and candidate sets are relabeled by the same permutation, the minimizing removal-candidate set and maximizing increase-candidate set are permuted accordingly. Hence, for unique selections,

$$i_c^{-\prime} = \sigma(i_c^-), \qquad i_c^{+\prime} = \sigma(i_c^+).$$

Removing or introducing these corresponding units therefore produces adjacent-rank representations related by the same matched relabeling.

The remaining quantities used by the rank-update rule are preserved as well. The coordinate-local reference μ_c is symmetric over the current rank units, and matched candidate calibration gives

$$\bar{U}'_{c,\sigma(i_c^+)} = \bar{U}_{c,i_c^+}.$$

Consequently, q_c^-, q_c^+, and their threshold-relative margins are unchanged, so a unique eligible update is identical up to the same relabeling. This proves Proposition 1.

The claim is restricted to matched permutations of latent units; general invertible changes of Tucker latent basis alter the individual directions and are outside this unit-wise relabeling setting. Under exact ties, the minimizing or maximizing candidate sets remain equivariant, whereas Eq. (B1) applies only to unique selections. The implementation resolves exact ties deterministically by physical index and therefore makes no single-candidate invariance claim in the non-unique case.

Table B1: Consistency under matched latent relabelings. Immediate agreement compares the next rank update from the same trained state and directly evaluates the one-step consistency property. Continued agreement is measured after subsequent training and additionally reflects optimization-path divergence.

Decomp.	Structural selection	Rank evidence	Output error	Immediate agr.	Continued agr.	Final-rank std.
Tucker	Physical-index	Index only	$< 10^{-6}$	63%	61%	0.72
	Physical-index	Task evidence	$< 10^{-6}$	76%	74%	0.51
	Evidence-based	Task evidence	$< 10^{-6}$	**100%**	**96%**	**0.09**
CP	Physical-index	Index only	$< 10^{-6}$	66%	64%	0.68
	Physical-index	Task evidence	$< 10^{-6}$	79%	77%	0.46
	Evidence-based	Task evidence	$< 10^{-6}$	**100%**	**95%**	**0.11**

B.2 Empirical Check under Matched Relabeling

We empirically verify the one-step property by starting from the same trained checkpoint and applying matched permutations to Tucker directions or CP components that preserve the represented tensor. We then compare the immediate next rank update under the original and relabeled representations. For reference, we also consider physical-index selection, which removes the largest-index current unit and introduces the smallest-index available candidate while otherwise retaining the same rank-learning protocol. We additionally report agreement after continued training to assess how the initial structural choice propagates along the subsequent optimization path.

All matched relabelings preserve the model output within 10^{-6}. Physical-index selection remains label-dependent even when task evidence is used, with immediate agreement of 76% for Tucker and 79% for CP. Evidence-based selection instead achieves 100% immediate agreement for both decompositions, consistent with Proposition 1. After continued training, agreement remains 96% for Tucker and 95% for CP, with final-rank standard deviations of 0.09 and 0.11, respectively. These continued-training results additionally reflect optimization-path divergence and are therefore not interpreted as an exact test of the one-step consistency property.

C Additional Experimental Results

C.1 Super-Resolution Results

Comparison with representative lightweight methods. Table C1 provides task-level context by comparing TGRank with representative lightweight $\times 4$ super-resolution methods. These results complement the controlled rank-selection comparison in Table 1, where the rank-selection methods share the same representation pathway and training protocol.

Table C1: Comparison with representative lightweight $\times 4$ super-resolution methods. This task-level comparison complements the controlled rank-selection study in Table 1. External method results are taken from their cited sources; the MambaIRv2-Light and TGRank–Tucker rows use our matched evaluation. Corresponding cross-scale results at $\times 2$ and $\times 3$ are reported in Table D9.

Method	Params (K)	MACs (G)	Set5	Set14	B100	Urban100	Manga109
SwinIR-Light (Liang et al., 2021)	930	31.8	32.44/0.8976	28.77/0.7858	27.69/0.7406	26.48/0.7980	30.92/0.9151
ASSANet (Pan et al., 2026)	456	26.0	32.57/0.8985	28.88/0.7877	27.76/0.7428	26.66/0.8015	31.28/0.9153
MaIR (Li et al., 2025)	1374	136.6	32.62/0.8998	28.90/0.7882	27.77/0.7431	26.73/0.8049	31.34/0.9183
MambaIR (Guo et al., 2024)	1383	143.0	32.51/0.8993	28.85/0.7876	27.75/0.7423	26.75/0.8051	31.26/0.9175
MambaIRv2-Light (Guo et al., 2025)	790	75.6	32.51/0.8992	28.84/0.7878	27.75/0.7426	26.82/0.8079	31.24/0.9182
TGRank–Tucker (Ours)	856	90.4	32.68/0.9011	29.01/0.7908	27.82/0.7447	27.12/0.8151	31.64/0.9213

Complete three-seed statistics. Tables C2 and C3 report the complete three-seed PSNR and SSIM statistics for the controlled super-resolution comparisons summarized in Table 1. Figure C1 provides a complementary qualitative comparison for representative benchmark images.

Table C2: Three-seed PSNR statistics for the controlled $\times 4$ super-resolution comparison in Table 1.

Method	Set5	Set14	B100	Urban100	Manga109
MambaIRv2-Light	32.51±0.02	28.84±0.02	27.75±0.01	26.82±0.02	31.24±0.03
Fixed Tucker $(8, 8, 4)$	32.53±0.01	28.86±0.01	27.76±0.01	26.86±0.02	31.30±0.02
Fixed Tucker $(12, 12, 6)$	32.57±0.02	28.90±0.01	27.78±0.01	26.92±0.02	31.39±0.02
Manual nonuniform	32.53±0.02	28.87±0.02	27.76±0.01	26.87±0.02	31.32±0.02
HOSVD-based	32.51±0.02	28.85±0.02	27.75±0.01	26.84±0.02	31.27±0.02
MARS-based	32.58±0.02	28.91±0.02	27.78±0.01	26.93±0.02	31.40±0.02
FlexLoRA-based	32.59±0.02	28.92±0.02	27.79±0.01	26.95±0.02	31.43±0.02
TGRank–Tucker	32.68±0.01	29.01±0.01	27.82±0.01	27.12±0.01	31.64±0.02
Fixed CP, rank 8	32.54±0.02	28.87±0.02	27.76±0.01	26.88±0.02	31.33±0.03
TGRank–CP	32.70±0.02	29.01±0.01	27.82±0.01	27.04±0.02	31.62±0.03

Table C3: Three-seed SSIM statistics for the controlled $\times 4$ super-resolution comparison in Table 1.

Method	Set5	Set14	B100	Urban100	Manga109
MambaIRv2-Light	0.8992±0.0002	0.7878±0.0002	0.7426±0.0001	0.8079±0.0002	0.9182±0.0003
Fixed CP, $(R = 8)$	0.8994±0.0002	0.7883±0.0002	0.7430±0.0001	0.8095±0.0002	0.9189±0.0003
Fixed Tucker $(8, 8, 4)$	0.8993±0.0001	0.7881±0.0001	0.7428±0.0001	0.8091±0.0002	0.9186±0.0002
Fixed Tucker $(12, 12, 6)$	0.8997±0.0002	0.7888±0.0001	0.7433±0.0001	0.8104±0.0002	0.9192±0.0002
Manual nonuniform	0.8994±0.0002	0.7884±0.0002	0.7430±0.0001	0.8094±0.0002	0.9188±0.0002
HOSVD-based	0.8991±0.0002	0.7879±0.0002	0.7426±0.0001	0.8087±0.0002	0.9183±0.0002
MARS-based	0.8998±0.0002	0.7890±0.0002	0.7434±0.0001	0.8107±0.0002	0.9194±0.0002
FlexLoRA-based	0.8999±0.0002	0.7892±0.0002	0.7437±0.0001	0.8113±0.0002	0.9197±0.0002
TGRank–CP	0.9006±0.0002	0.7906±0.0001	0.7445±0.0001	0.8139±0.0002	0.9207±0.0003
TGRank–Tucker	0.9011±0.0001	0.7908±0.0001	0.7447±0.0001	0.8151±0.0001	0.9213±0.0002

C.2 CIFAR-100 Results by Seed

Table C4 reports per-seed CIFAR-100 accuracy for the backbone and the adaptive Tucker/CP variants. These per-seed entries are paired with the same-seed backbone measurements; matched fixed-rank comparisons are reported only in aggregate in Table 2.

Table C4: Per-seed CIFAR-100 accuracy at epoch 200. The final row reports mean±standard deviation over three paired seeds.

	Top-1			Top-5		
Seed	ResNet-18	TGRank–Tucker	TGRank–CP	ResNet-18	TGRank–Tucker	TGRank–CP
10	75.50	76.27	75.93	92.70	92.96	92.87
20	75.66	76.38	76.06	92.94	93.07	93.00
30	75.95	76.43	76.13	93.10	93.12	93.07
Mean±std.	75.70 ± 0.23	76.36 ± 0.08	76.04 ± 0.10	92.91 ± 0.20	93.05 ± 0.08	92.98 ± 0.10

C.3 Final Ranks and Update Accounting

Table C5 reports the final CIFAR-100 ranks and update counts. All applied rank increases are reversed before the ranks are frozen, so the final adaptation is dominated by rank decrease. Retained rank growth is examined separately in the low-initialization super-resolution and ADE20K settings.

C.4 Semantic-Segmentation Results

Table C6 expands the aggregate ADE20K results in Table 3 with per-seed metrics, final Tucker ranks, and rank-update accounting for both initializations. An increase is counted as retained if it remains in the learned rank structure when adaptation ends; an increase that is later removed is counted as reversed.

Figure C2 provides qualitative examples from the ADE20K validation set. These examples illustrate representative differences around object boundaries and thin structures, while the quantitative comparison is based on the complete validation set reported in Table 3 and Table C6.

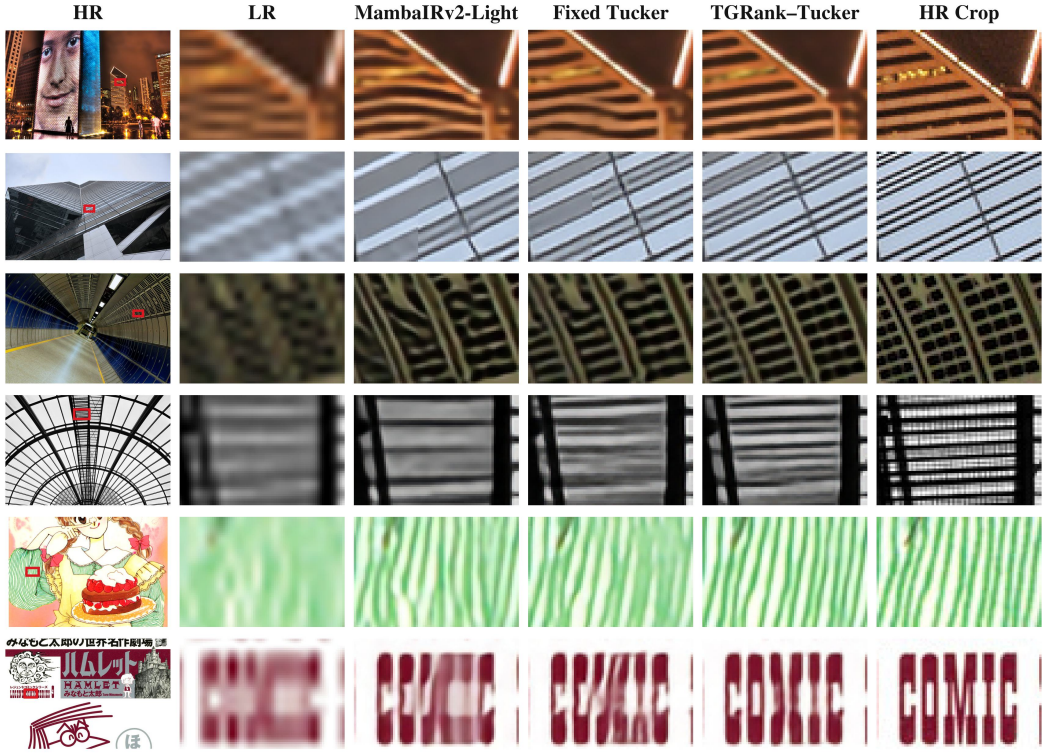


Figure C1: Qualitative comparison for $\times 4$ super-resolution on representative Urban100 and Manga109. From left to right: high-resolution reference (HR), the low-resolution input (LR), MambaIRv2-Light, Fixed Tucker, TGRank–Tucker, and the cropped HR reference. Red boxes mark the regions enlarged in the remaining columns. The examples complement the quantitative benchmark statistics and illustrate reconstruction of fine structures.

Table C5: Per-seed final CIFAR-100 ranks and update counts. Dec., Inc., and Rev. denote applied rank decreases, applied rank increases, and increases that are later reversed, respectively. All final ranks lie within the stage-wise admissible bounds in Section A.2.

Decomp.	Seed	Final stage ranks	Σr	Dec./Inc./Rev.
Tucker	10	$(7,7,3)\,\|\,(2,2,2)\,\|\,(2,2,3)\,\|\,(2,2,2)$	36	18/6/6
Tucker	20	$(4,4,2)\,\|\,(4,4,3)\,\|\,(2,2,2)\,\|\,(2,2,2)$	33	17/2/2
Tucker	30	$(7,7,4)\,\|\,(3,4,2)\,\|\,(1,1,4)\,\|\,(2,2,2)$	39	11/2/2
CP	10	$4\|2\|2\|2$	10	13/1/1
CP	20	$4\|2\|3\|4$	13	12/3/3
CP	30	$7\|4\|2\|3$	16	7/1/1

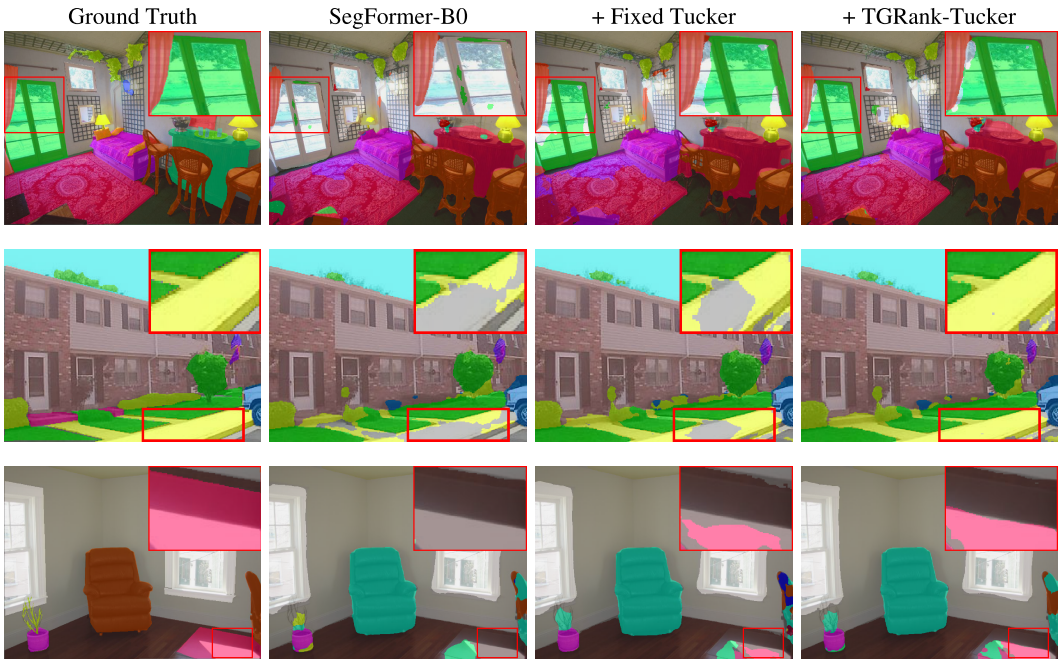


Figure C2: Qualitative comparison on ADE20K. From left to right: ground truth, SegFormer-B0, SegFormer-B0 with fixed Tucker ranks, and SegFormer-B0 with TGRank–Tucker. Red boxes and enlarged regions highlight representative differences around semantic boundaries and thin structures.

Table C6: Per-seed ADE20K results and final Tucker ranks. Aggregate rows report mean±standard deviation over the three paired seeds and correspond to the results in Table 3. For TGRank, retained increases are rank additions that remain when rank adaptation ends and through the subsequent fixed-rank refinement; reversed increases are additions removed before adaptation ends.

Setting	Seed	Final ranks	Σr	mIoU	mAcc	Ret./rev. inc.
SegFormer-B0	10	–	–	37.29	47.41	–
	20	–	–	37.43	47.55	–
	30	–	–	37.48	47.62	–
	Mean±std.	–	–	37.40 ± 0.10	47.53 ± 0.11	–
Fixed Tucker	10	$(8,8,4) \mid (8,8,4) \mid (8,8,4) \mid (8,8,4)$	80	37.40	47.56	–
	20	$(8,8,4) \mid (8,8,4) \mid (8,8,4) \mid (8,8,4)$	80	37.53	47.68	–
	30	$(8,8,4) \mid (8,8,4) \mid (8,8,4) \mid (8,8,4)$	80	37.57	47.74	–
	Mean±std.	–	80	37.50 ± 0.09	47.66 ± 0.09	–
Fixed Tucker, low init.	10	$(4,4,2) \mid (4,4,2) \mid (4,4,2) \mid (4,4,2)$	40	37.18	47.30	–
	20	$(4,4,2) \mid (4,4,2) \mid (4,4,2) \mid (4,4,2)$	40	37.30	47.43	–
	30	$(4,4,2) \mid (4,4,2) \mid (4,4,2) \mid (4,4,2)$	40	37.36	47.50	–
	Mean±std.	–	40	37.28 ± 0.09	47.41 ± 0.10	–
TGRank, standard init.	10	$(8,7,4) \mid (5,5,3) \mid (9,8,5) \mid (5,4,2)$	65	37.81	48.91	5/24
	20	$(8,7,4) \mid (6,6,3) \mid (9,8,5) \mid (5,4,2)$	67	37.88	49.00	6/27
	30	$(8,7,4) \mid (6,5,3) \mid (9,8,5) \mid (5,4,2)$	66	37.95	49.09	7/30
	Mean±std.	–	66 ± 1	37.88 ± 0.07	49.00 ± 0.09	$6 \pm 1/27 \pm 3$
TGRank, low init.	10	$(6,5,3) \mid (5,4,3) \mid (8,7,5) \mid (5,4,2)$	57	37.51	47.68	34/17
	20	$(6,5,3) \mid (5,5,3) \mid (8,8,5) \mid (5,4,2)$	59	37.63	47.81	36/20
	30	$(6,5,3) \mid (5,5,3) \mid (8,7,5) \mid (5,4,2)$	58	37.69	47.88	38/23
	Mean±std.	–	58 ± 1	37.61 ± 0.09	47.79 ± 0.10	$36 \pm 2/20 \pm 3$

D Additional Analysis

D.1 Fixed-Rank Response Analysis

Figure 1 summarizes the mode-3 fixed-rank responses used to motivate rank learning. Table D1 reports the corresponding DIV2K-validation PSNR values, while Table D2 extends the saturation analysis to all three Tucker modes across the four representation stages. Table D3 further examines the sensitivity of the mode-3 saturation ranks to the tolerance δ_{sat}. All results follow the controlled $\times 4$ super-resolution retraining protocol defined in Appendix A.4 and are used only as offline diagnostics. Together, these results show that the site-dependent saturation pattern is not specific to a single tolerance: the mode-3 rank remains unchanged at Stage 0, decreases at Stages 1 and 3, and increases at Stage 2.

Table D1: DIV2K-validation PSNR underlying the mode-3 fixed-rank responses in Figure 1 for the controlled $\times 4$ super-resolution study. Results are reported as mean $\pm$ 95% confidence interval over five paired seeds. Bold marks r_{sat} at $\delta_{\mathrm{sat}} = 0.02$ dB; the saturation decision uses the paired performance-deficit interval defined in Appendix A.4.

Coordinate	$r = 2$	$r = 3$	$r = 4$	$r = 5$	$r = 6$
Stage 0, mode 3	27.02±0.03	27.12±0.02	**27.18±0.01**	27.19±0.01	27.18±0.02
Stage 1, mode 3	27.10±0.02	**27.18±0.01**	27.18±0.02	27.18±0.02	27.17±0.02
Stage 2, mode 3	26.97±0.03	27.03±0.02	27.13±0.02	**27.20±0.01**	27.20±0.02
Stage 3, mode 3	**27.17±0.01**	27.17±0.02	27.16±0.02	27.15±0.02	27.14±0.03

Table D2: Tucker saturation ranks across stages and modes for the controlled $\times 4$ super-resolution study on DIV2K validation. This extends the mode-3 rank-response analysis in Figure 1 and Table D1 to all three Tucker modes. Saturation ranks are determined using the protocol in Appendix A.4.

Coordinate	Stage 0	Stage 1	Stage 2	Stage 3
Mode 1	8	7	5	10
Mode 2	8	7	5	8
Mode 3	4	3	5	2

Table D3: Sensitivity of the mode-3 Tucker saturation ranks to δ_{sat} for the controlled $\times 4$ super-resolution study on DIV2K validation. The saturation-rank pattern from Figure 1 and Table D1 remains unchanged across all tested tolerances. Relative to the initial mode-3 rank of 4, the four sites correspond to unchanged, lower, higher, and lower saturation ranks, respectively.

Coordinate	$\delta_{\mathrm{sat}} = 0.01$ dB	0.02 dB	0.03 dB
Stage 0, mode 3	4	4	4
Stage 1, mode 3	3	3	3
Stage 2, mode 3	5	5	5
Stage 3, mode 3	2	2	2

D.2 Learned Rank Structure and Rank-Learning Path

We further examine whether the task-performance gains arise from the rank structure discovered by TGRank, from optimization along the changing-rank training path, or from both. Table D4 compares three settings. *Initial-rank fixed* trains the prescribed initial ranks without rank updates. *Learned-rank fixed* retrains the ranks discovered by TGRank from the matched initialization with rank learning disabled. *Adaptive* denotes the complete TGRank training run.

Across both Tucker and CP representations, retraining the discovered ranks improves over the corresponding initial-rank fixed model, showing that the learned rank structure retains task value independently of the rank-learning trajectory. The complete TGRank run provides a further improvement over learned-rank fixed retraining, indicating that optimization along the changing-rank path also

contributes to final task performance. Thus, the benefit of TGRank is not explained solely by either the final rank structure or the training path through which it is reached.

Table D4: Separating the learned rank structure from the rank-learning path. *Initial-rank fixed* trains the prescribed initial ranks without rank updates. *Learned-rank fixed* retrains the ranks discovered by TGRank with rank learning disabled. *Adaptive* denotes the complete TGRank training run. Results are reported as mean±standard deviation over three paired seeds.

Task	Decomp.	Initial-rank fixed	Learned-rank fixed	Adaptive
CIFAR Top-1	Tucker	75.73±0.16	76.08±0.10	76.36±0.08
	CP	75.48±0.20	75.79±0.12	76.04±0.10
Urban100 PSNR	Tucker	26.86±0.02	27.05±0.02	27.12±0.01
	CP	26.88±0.02	26.99±0.02	27.04±0.02

D.3 Effect of Initialization on Rank Learning

Table D5: Effect of Tucker initialization on rank learning for $\times 4$ super-resolution on Urban100. Retained and reversed increases are reported separately to distinguish rank growth retained in the learned structure from temporary increases that are subsequently removed.

Init.	Σr: init.→final	Retained/reversed increases	Fixed / Adaptive PSNR
$(4, 4, 2)$	$40 \rightarrow 76\pm2$	$36\pm2/20\pm3$	26.72/27.08
$(8, 8, 4)$	$80 \rightarrow 77\pm3$	$14\pm1/27\pm3$	26.86/27.12
$(12, 12, 6)$	$120 \rightarrow 82\pm3$	$2\pm1/18\pm2$	26.92/27.09

We next examine how rank learning depends on the initial representation capacity. Table D5 compares low, standard, and high Tucker initializations under the same $\times 4$ super-resolution setting. The corresponding initial rank sums are 40, 80, and 120, whereas the learned rank sums are 76 ± 2, 77 ± 3, and 82 ± 3, respectively. Thus, widely different initial capacities lead to substantially closer final rank scales.

The direction of rank learning also changes with initialization. Starting from the low initialization produces substantial retained rank growth, whereas the high initialization is dominated by rank decrease. The standard initialization exhibits both upward and downward changes before reaching its final structure. Despite these different rank-learning paths, final Urban100 performance varies by only 0.04 dB across the three initializations.

Together, these results show that TGRank does not impose a predetermined compression or expansion direction. Instead, the initial capacity affects the path through which rank learning proceeds, while the final task performance and learned rank scale are substantially less sensitive to the starting rank. Figure 3 complements this analysis by showing the stage–mode rank trajectories for the standard initialization.

D.4 Rank-Update Stability and Component Analysis

Table D6 evaluates rank-update stability rather than additive attribution of task-performance gains. Random adjacent updates, shared rank evidence, and decrease-only adaptation provide coarse controls. The progressive variants then introduce evidence-based removal selection, candidate selection by $\bar{P}$, calibrated utility U, and the complete temporal stabilization rule.

Across the progressive bidirectional variants, reversed increases decrease from 35 ± 3 to 27 ± 3 while the total number of applied changes remains near 85 and PSNR improves modestly. These results indicate that the components primarily improve the reliability of rank updates rather than acting as independent sources of task-performance gain.

Table D6: Rank-update stability and component analysis on Urban100 $\times 4$. Applied is the total number of applied adjacent rank changes, and Rev. inc. counts rank increases that are later reversed. The comparison focuses on rank-update stability rather than attributing task-performance gains to individual components.

Variant	Applied	Rev. inc.	PSNR
Random adjacent updates	85.0 ± 2.4	38.0 ± 3.0	27.02 ± 0.03
Shared rank evidence	85.0 ± 2.0	36.0 ± 3.0	27.05 ± 0.02
Decrease only	38.0 ± 1.6	0	27.06 ± 0.02
Physical-index structural selection	86.0 ± 2.0	35.0 ± 3.0	27.04 ± 0.02
Evidence-based removal selection	85.0 ± 2.0	33.0 ± 3.0	27.06 ± 0.02
+ increase-candidate selection by P	85.0 ± 2.0	31.0 ± 3.0	27.08 ± 0.02
+ calibrated utility U	85.0 ± 2.0	29.0 ± 3.0	27.10 ± 0.02
TGRank (full update rule)	$\mathbf{85.0\pm1.0}$	$\mathbf{27.0\pm3.0}$	$\mathbf{27.12\pm0.01}$

Table D7: Sensitivity to the rank-decrease and rank-increase support thresholds on Urban100 $\times 4$. The default thresholds are $(\tau_-^0, \tau_+^0) = (0.35, 0.002)$, and α_- and α_+ independently scale the corresponding defaults. All remaining rank-update settings are fixed. Results are reported for seed 10.

α_-	α_+	PSNR	Final Σr	Dec./Inc.
0.8	0.8	27.13	81	41/42
0.8	1.0	27.14	80	42/42
0.8	1.2	27.13	79	43/42
1.0	0.8	27.13	79	43/42
1.0	**1.0**	**27.12**	**77**	**44/41**
1.0	1.2	27.13	76	45/41
1.2	0.8	27.11	77	45/42
1.2	1.0	27.12	75	46/41
1.2	1.2	27.11	74	47/41

D.5 Threshold and Normalization Sensitivity

We first examine sensitivity to the support thresholds used for rank decrease and rank increase. The default values are $\tau_-^0 = 0.35$ and $\tau_+^0 = 0.002$, shared across the tasks and factorizations considered in the main experiments. Starting from these defaults, we independently vary $\tau_- = \alpha_- \tau_-^0$ and $\tau_+ = \alpha_+ \tau_+^0$, with $(\alpha_-, \alpha_+) \in \{0.8, 1.0, 1.2\}^2$, while keeping the remaining rank-update settings fixed.

As shown in Table D7, both rank directions remain active throughout the tested range, while Urban100 PSNR varies by only 0.03 dB. Increasing α_- makes the rank-decrease condition $q_c^- < \tau_-$ less restrictive and consequently produces more decreases, whereas increasing α_+ makes the rank-increase condition $q_c^+ > \tau_+$ more selective. Accordingly, the final rank sum varies from 74 to 81 across the tested settings, while task performance remains stable. These results indicate that the thresholds control the aggressiveness of structural adaptation without requiring precise tuning for task performance within the tested range.

We additionally compare unnormalized, global, and coordinate-local evidence normalization. As shown in Table D8, normalization changes the learned rank structure more noticeably than final task performance; coordinate-local normalization is used to align evidence scales across rank coordinates in the default method.

D.6 Cross-Scale Super-Resolution Evaluation

We further evaluate TGRank at $\times 2$ and $\times 3$ super-resolution to examine whether the rank-learning behavior observed in the primary $\times 4$ setting transfers across reconstruction scales. Table D9 provides task-level context using representative lightweight SR methods and matched comparisons between fixed-rank and TGRank representations. External method results are taken from their cited

Table D8: Sensitivity to evidence normalization on Urban100 $\times 4$. Final-rank agreement and mean $|\Delta r_c|$ are measured relative to the default coordinate-local normalization. Final-rank agreement is the fraction of rank coordinates whose final values match those obtained with coordinate-local normalization.

Normalization	Final-rank agreement	Mean $\lvert\Delta r_c\rvert$	PSNR
None	$50\pm8\%$	0.83 ± 0.18	27.13 ± 0.03
Global	$67\pm7\%$	0.50 ± 0.14	27.16 ± 0.02
Coordinate-local (default)	–	–	27.12 ± 0.01

Table D9: Cross-scale super-resolution evaluation at $\times 2$ and $\times 3$. External lightweight-method results are taken from the sources cited in Table C1. Our fixed and TGRank variants use matched representation structures and training protocols; results are reported for seed 10. Rank sums are compared only within each factorization family.

Method	Σr: init.$\rightarrow$final	Set5	Set14	B100	Urban100	Manga109
$\times 2$ *super-resolution*						
SwinIR-Light	–	38.14/0.9611	33.86/0.9206	32.31/0.9012	32.76/0.9340	39.12/0.9783
ASSANet	–	38.23/0.9613	34.04/0.9205	32.31/0.9017	32.97/0.9349	39.50/0.9789
MaIR	–	38.20/0.9611	33.91/0.9209	32.34/0.9016	32.97/0.9359	39.32/0.9779
MambaIR	–	38.16/0.9610	34.00/0.9212	32.34/0.9017	32.92/0.9356	39.31/0.9779
MambaIRv2-Light	–	38.26/0.9615	34.09/0.9221	32.36/0.9019	33.26/0.9378	39.35/0.9785
Fixed CP	$32 \rightarrow 32$	38.27/0.9614	34.13/0.9229	32.37/0.9021	33.29/0.9382	39.41/0.9786
TGRank–CP	$32 \rightarrow 16$	38.35/0.9618	34.29/0.9245	32.43/0.9029	33.53/0.9402	39.64/0.9789
Fixed Tucker	$80 \rightarrow 80$	38.28/0.9615	34.15/0.9231	32.37/0.9022	33.30/0.9384	39.43/0.9787
TGRank–Tucker	$80 \rightarrow 69$	38.37/0.9619	34.38/0.9248	32.43/0.9029	33.53/0.9403	39.66/0.9792
$\times 3$ *super-resolution*						
SwinIR-Light	–	34.62/0.9289	30.54/0.8463	29.20/0.8082	28.66/0.8624	33.98/0.9478
ASSANet	–	34.70/0.9292	30.51/0.8456	29.27/0.8101	28.85/0.8649	34.54/0.9498
MaIR	–	34.75/0.9300	30.63/0.8479	29.29/0.8103	28.92/0.8676	34.46/0.9497
MambaIR	–	34.72/0.9296	30.63/0.8475	29.29/0.8099	29.00/0.8689	34.39/0.9495
MambaIRv2-Light	–	34.71/0.9298	30.68/0.8483	29.26/0.8098	29.01/0.8689	34.41/0.9497
Fixed CP	$32 \rightarrow 32$	34.73/0.9298	30.68/0.8483	29.28/0.8103	29.06/0.8697	34.47/0.9497
TGRank–CP	$32 \rightarrow 19$	34.85/0.9306	30.77/0.8500	29.35/0.8116	29.35/0.8748	34.75/0.9513
Fixed Tucker	$80 \rightarrow 80$	34.74/0.9299	30.69/0.8485	29.28/0.8104	29.08/0.8700	34.50/0.9499
TGRank–Tucker	$80 \rightarrow 77$	34.85/0.9307	30.76/0.8499	29.35/0.8115	29.34/0.8747	34.80/0.9515

sources, whereas our fixed and TGRank variants use the same representation structure and training protocol within each scale. Our cross-scale results are reported for seed 10.

TGRank learns different rank budgets across scales: the final Tucker rank sums are 69 and 77 at $\times 2$ and $\times 3$, while the corresponding CP component counts are 16 and 19. At both scales, TGRank improves over its matched fixed-rank counterpart across the five benchmarks. These results indicate that the learned rank structure adapts to reconstruction scale rather than reusing a fixed capacity allocation.

E Computational Cost and Fixed-Rank Export

Table E1 reports training and deployment cost under a common hardware environment. Rank evaluation is invoked sparsely relative to ordinary task training; consequently, candidate calibration increases the cost of individual scheduled evaluations but adds only 0.9% to end-to-end training wall time.

For deployment, the learned representation is exported as a conventional fixed-rank model by retaining the selected Tucker factor directions and matched core slices, or the selected CP components. Parameter counts, MACs, and latency are measured on this exported graph, whose output is numerically equivalent to the training-time representation.

Table E1: Training and deployment cost measured on the same hardware. Iteration time measures ordinary task training, whereas rank-evaluation cost is incurred only at scheduled evaluations. Wall Δ denotes the end-to-end training overhead relative to the matched fixed-rank baseline. Deployment quantities are measured after fixed-rank export.

Method	Iter. (ms)	Rank eval. (s)	Wall Δ	Peak GiB	Rel. MACs	Lat. (ms)
Fixed $(8, 8, 4)$	802 ± 6	–	–	10.4	$1.00\times$	35.1
Candidate screening only (P)	804 ± 7	1.9 ± 0.1	$+0.3\%$	10.8	$0.99\times$	34.9
TGRank	804 ± 7	29.6 ± 1.4	$+0.9\%$	11.6	$0.99\times$	34.9

F Implementation Verification

We verify four invariants directly implied by the rank-update procedure: zero-contribution insertion, restoration after temporary calibration, evidence–update identity, and separation between calibration and utility evaluation. Table F1 summarizes these checks. Numerical equivalence and state restoration are verified to a maximum absolute error of 10^{-6}.

Table F1: Implementation invariants for TGRank rank updates. These checks verify the structural, state, selection, and evaluation-isolation properties required by the update procedure.

Check	Verified behavior
Zero-contribution insertion	Introducing an increase candidate from zero contribution preserves the represented output up to numerical precision.
State restoration	Temporary candidate calibration is fully discarded when no rank increase is applied, and checkpoint resumption reproduces the same rank-update state.
Evidence–update identity	The unit selected by the decrease or increase evidence is exactly the unit removed or introduced by the corresponding rank update.
Evaluation isolation	Candidate utility is evaluated on data disjoint from the calibration batch.