Exploring Tensor Product Attention in Vision Transformers under Limited Data and Compute: project report

Qiwu Wen [1]

Abstract

Tensor-product attention (TPA) reduces inference-time key/value (KV) storage by factorizing per-token $Q/K/V$ into low-rank components. We transfer vanilla TPA to small Vision Transformers and use image classification as a low-cost testbed to probe its vision-domain behavior, where we observe a clear accuracy gap to standard multi-head attention (MHA) in the small-h, small-d_h regime. Motivated by a rank-induced linear-span constraint in vanilla TPA, we propose *NonlinearTPA*, which injects a lightweight GELU-MLP on per-head vectors while keeping the TPA rank configuration unchanged, preserving the KV-friendly structure. Across CIFAR-10 and CIFAR-100 on a ViT-Tiny backbone, NonlinearTPA consistently improves vanilla TPA by a few top-1 points under a matched MLP parameter budget, with the best performance obtained by applying the MLP to all of $Q/K/V$.

1. Introduction

A key motivation of recent attention parameterizations is to reduce the storage of key/value (KV) representations in inference-time settings. Tensor-product attention (TPA) (Zhang et al., 2025) addresses this by factorizing per-token $Q/K/V$ slices into low-rank components, enabling a KV-friendly representation that can be substantially cheaper to store than standard multi-head attention (MHA).

Why study TPA in vision classification? TPA is motivated by KV storage in inference-time workloads, whereas standard image classification does not require an explicit KV cache at inference. In this work, we use vision classification as a controlled and low-cost testbed to probe whether the TPA parameterization generalizes to Vision Transformer backbones. Although the task differs from KV-cache-heavy workloads, the same factorization mechanism can be analyzed via a normalized KV-storage proxy (Sec. 3). Our goal is not to claim immediate gains on large-scale multimodal or video-generation systems, but to assess whether TPA can retain competitive representation capacity in small vision transformers and to expose performance–cost trends that may inform KV-sensitive vision applications.

Challenge in small ViTs. In Sec. 3, we analyze vanilla TPA in small vision transformers (e.g., ViT-Tiny) where the number of heads h and per-head dimension d_h are limited. Under KV-budgeted rank settings, we observe a clear performance gap between vanilla TPA and MHA, even when TPA uses more parameters (Table 1). This raises the following question: *can we improve the expressivity of TPA in small ViTs while keeping its KV-friendly factorized structure (i.e., without increasing R_K and R_V)?*

Approach: NonlinearTPA. We propose *NonlinearTPA*, a lightweight modification to vanilla TPA that injects a simple GELU-MLP on per-head vectors (Sec. 4). The design is motivated by a rank-induced linear-span constraint in vanilla TPA: per-head $q_{t,i}$ (and similarly $k_{t,i}, v_{t,i}$) lies in an at-most rank-dimensional subspace determined by TPA factors, limiting expressivity unless ranks are increased. NonlinearTPA aims to boost representation capacity while keeping the TPA rank configuration unchanged, and we study practical design choices such as MLP placement (on Q, K, V, or combinations) and a head-wise variant.

Results and contributions. On ViT-Tiny with fixed ranks $(R_q, R_k, R_v) = (16, 2, 2)$, NonlinearTPA consistently improves vanilla TPA on CIFAR-10 and CIFAR-100 (Table 2). Under a nearly matched MLP parameter budget across placements, applying the MLP to all of $Q/K/V$ performs best, improving top-1 validation accuracy by $+2.26$ points on CIFAR-10 and $+3.15$ points on CIFAR-100 over vanilla TPA. We also observe a preliminary indication that head-wise nonlinearities can provide additional gains in this small-h regime, though the effect requires more systematic validation (Table 3). Our contributions are:

- We analyze vanilla TPA in small vision transformers

[1]DIRO, Université de Montréeal, Montréeal, QC, Canada. Correspondence to: Qiwu Wen <qiwu.wen@umontreal.ca>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

and identify a clear performance–cost limitation under KV-budgeted rank settings (Sec. 3).

- We propose *NonlinearTPA*, a simple and parameter-efficient nonlinearity injection that targets the rank-induced expressivity constraint while keeping the TPA rank configuration unchanged (Sec. 4).
- We demonstrate consistent improvements over vanilla TPA on CIFAR-10/100, and conduct ablations over MLP placement and a head-wise variant under a matched parameter budget (Sec. 5).

2. Background and Related Work

2.1. Tensor Product Attention (TPA)

Motivation: KV-cache as an inference bottleneck. In autoregressive decoding, Transformers cache the key/value tensors from all past tokens to avoid recomputation. For standard multi-head attention (MHA), each token stores $K_t, V_t \in \mathbb{R}^{h \times d_h}$, leading to a per-token KV-cache cost of $2hd_h$ numbers (and linear growth with sequence length). Zhang et al. (2025) proposes Tensor Product Attention (TPA) to reduce this KV-cache footprint via a contextual low-rank factorization of Q, K, and V. (Zhang et al., 2025)

Contextual tensor-product factorization of Q, K, V. Let $x_t \in \mathbb{R}^{d_{\text{model}}}$ be the hidden state of token t. TPA constructs the per-token slices $Q_t, K_t, V_t \in \mathbb{R}^{h \times d_h}$ as sums of rank-1 outer products:

$$\begin{aligned} Q_t &= \frac{1}{R_Q} \sum_{r=1}^{R_Q} a_r^Q(x_t) \otimes b_r^Q(x_t), \\ K_t &= \frac{1}{R_K} \sum_{r=1}^{R_K} a_r^K(x_t) \otimes b_r^K(x_t), \\ V_t &= \frac{1}{R_V} \sum_{r=1}^{R_V} a_r^V(x_t) \otimes b_r^V(x_t). \end{aligned} \tag{1}$$

where $a_r^{\cdot}(x_t) \in \mathbb{R}^h$ are *head-dimension* factors and $b_r^{\cdot}(x_t) \in \mathbb{R}^{d_h}$ are *token/channel-dimension* factors. (Zhang et al., 2025) Equivalently, stacking factors into matrices $A_Q(x_t) \in \mathbb{R}^{R_Q \times h}$ and $B_Q(x_t) \in \mathbb{R}^{R_Q \times d_h}$ (with rows $a_r^Q(x_t)$ and $b_r^Q(x_t)$) yields

$$Q_t = \frac{1}{R_Q} A_Q(x_t)^\top B_Q(x_t), \tag{2}$$

and analogously for K_t and V_t. (Zhang et al., 2025) In the vanilla (fully contextual) form, these factors are produced by linear maps of x_t (e.g., $a_r^Q(x_t) = W_{a,r}^Q x_t$ and $b_r^Q(x_t) = W_{b,r}^Q x_t$), typically implemented by a single large projection followed by reshaping. (Zhang et al., 2025)

Attention computation remains standard After forming Q, K, V, TPA applies the usual scaled dot-product attention head-wise, and then concatenates heads followed by an output projection, i.e., TPA is a drop-in replacement of MHA at the attention level. (Zhang et al., 2025)

KV-cache memory reduction The key practical gain of TPA appears at decoding time: instead of caching full K_t and V_t, TPA caches only their factor matrices (e.g., $A_K(x_t)$ and $B_K(x_t)$ for keys, and $A_V(x_t)$ and $B_V(x_t)$ for values). This yields a per-token KV-cache cost

$$\underbrace{R_K(h + d_h)}_{\text{for } K} + \underbrace{R_V(h + d_h)}_{\text{for } V} = (R_K + R_V)(h + d_h), \tag{3}$$

so the normalized ratio to MHA is

$$\frac{(R_K + R_V)(h + d_h)}{2hd_h}. \tag{4}$$

When $R_K, R_V \ll h$ and d_h is moderate (e.g., 64 or 128), this can provide large (often order-of-magnitude) KV-cache savings. (Zhang et al., 2025)

Scope and implementation choices. Beyond the basic factorization, Zhang et al. (2025) discusses several extensions and implementation aspects, such as compatibility with positional encoding choices (e.g., RoPE) and efficient algorithms for TPA-style attention (e.g., FlashTPA). In this work, we focus on the *vanilla* TPA parameterization as a drop-in replacement of MHA and study its representational behavior in small vision transformers. We do not use the additional engineering optimizations or positional-encoding-specific variants, as our goal is to isolate the expressivity limitations induced by the rank parameterization and evaluate lightweight architectural remedies.

2.2. Related Work

KV-efficient attention in Transformers. Standard multi-head attention (MHA) assigns each head its own key/value projections, leading to a per-token KV cache that scales with the number of heads (Vaswani et al., 2017). Multi-Query Attention (MQA) reduces decoding-time KV memory by sharing keys and values across heads (Shazeer, 2019), while Grouped-Query Attention (GQA) interpolates between MHA and MQA by sharing KV within head groups (Ainslie et al., 2023). Most relevant to our study, Tensor Product Attention (TPA) factorizes the *activations* (Q/K/V) via contextual tensor decompositions, substantially shrinking the KV cache and providing a unifying perspective in which MHA/MQA/GQA can be expressed as special non-contextual cases (Zhang et al., 2025). TPA is also designed to be compatible with rotary position embedding (RoPE) and comes with a specialized decoding procedure (FlashTPA) for efficient autoregressive inference (Zhang et al., 2025). In this project, we focus on *vanilla* TPA as a KV-friendly attention primitive and study its behavior when transferred to small vision transformers.

Efficient attention in vision transformers. Vision Transformers (ViTs) (Dosovitskiy et al., 2020) often face high cost when the number of visual tokens grows (e.g., high-resolution inputs). A large body of work therefore improves efficiency by restricting or approximating attention along the *token dimension*, e.g., windowed attention in Swin Transformer (Liu et al., 2021), spatial-reduction attention in Pyramid Vision Transformer (PVT) (Wang et al., 2021), and sparse/long-range variants such as Multi-Scale Vision Longformer (Zhang et al., 2021). Other approaches propose efficient attention/backbone designs that mitigate quadratic attention cost in vision models, such as the multi-scale linear attention in EfficientViT (Liu et al., 2023). These methods primarily target compute/memory scaling with the number of image tokens, whereas our work studies a complementary axis: *KV-factorized* attention originally motivated by LLM decoding, evaluated here on image classification as a low-cost testbed for vision-domain transfer.

Nonlinearity for alleviating low-rank bottlenecks (connection to TPA). Low-rank structure is widely used for parameter-efficient adaptation, notably LoRA, which represents weight updates as a linear combination of rank-r factors (Hu et al., 2022). A recurring observation in this line of work is that low-rank linearity may limit expressivity in practice, motivating *nonlinear* extensions that insert lightweight nonlinear mappings while retaining parameter efficiency (e.g., Deng et al. (2025); Dong et al. (2025)). Conceptually, vanilla TPA similarly represents contextual Q/K/V activations as sums of rank-1 outer products (Zhang et al., 2025), which likewise induces a rank-controlled linear-combination constraint (Sec. 4.1). This parallel motivates our approach: we inject a small MLP on top of TPA-produced per-head vectors to relax the rank-induced bottleneck, while keeping the underlying KV-factorized structure and rank hyperparameters unchanged.

3. Preliminary Observation: Vanilla TPA in Small ViTs

We first analyze vanilla tensor-product attention (TPA) in small vision transformers (e.g., ViT-Tiny), where the number of heads h and the per-head dimension d_h are relatively small ($D = h\,d_h$). Our goal is to understand how the rank hyperparameters (r_q, r_k, r_v) interact with this small-model regime, and how they constrain the performance–cost trade-off observed in Table 1.

KV storage cost in MHA vs. TPA A key motivation of TPA is to reduce the memory required to store key/value representations. For MHA, the per-token K/V storage scales as $\mathcal{O}(hd_h)$. In contrast, TPA factorizes the per-token $K_t, V_t \in \mathbb{R}^{h\times d_h}$ into rank-R_k and rank-R_v outer-product sums, and stores only the factors; this yields a per-token KV storage on the order of $\mathcal{O}\big((R_k + R_v)(h + d_h)\big)$, which can be substantially smaller when $R_k, R_v \ll \min(h, d_h)$.

Table 1. TPA–MHA trade-off on CIFAR-100 (best val acc). KV storage cost is normalized to MHA (=1.0). **Bold** indicates the best within each group.

Model	$R_q/R_k/R_v$	KV storage cost	#Params (M)	Best Val Acc
MHA(baseline)	–	1.000	5.544	**0.452**
TPA	16/2/2	0.698	7.313	**0.403**
TPA	16/1/1	0.349	7.003	0.380
TPA	8/2/2	0.698	6.072	0.397
TPA	8/1/1	0.349	5.761	0.390
TPA	4/2/2	0.698	5.451	0.401
TPA	4/1/1	0.349	5.141	0.388

Why the advantage shrinks in small ViTs In small ViTs such as ViT-Tiny, h and d_h are limited (e.g., h=3 and d_h=64 in our setup), leaving less room for KV-factorization to provide dramatic savings. For instance, even with R_k=R_v=1, the normalized KV cost is reduced but not by orders of magnitude, and increasing R_k, R_v quickly violates a tight KV budget. As a result, the feasible rank configurations for K/V are heavily constrained, which can limit the expressivity of vanilla TPA in this regime.

Implication: the remaining knob is often R_Q, but it is parameter-inefficient. A natural way to increase the expressivity of vanilla TPA is therefore to raise the query rank R_Q. However, this knob is parameter-inefficient in small models. Following Zhang et al. (2025), the TPA parameter count scales as

$$\#\text{Params} \propto d_{\text{model}}(R_Q{+}R_K{+}R_V)(h{+}d_h) + d_{\text{model}}\,h\,d_h$$

so increasing R_Q leads to a near-linear increase in parameters. Empirically, increasing R_Q yields limited gains under the same training recipe, while significantly increasing the parameter count (Table 1).

Design goal These observations suggest a clear target: *retain the KV-cost advantage of TPA (by keeping R_K and R_V small), while closing the performance gap to MHA as much as possible* in the small-h, small-d_h regime—rather than aiming to surpass MHA.

4. Method

Motivated by the preliminary observation in Sec. 3, we aim to improve the expressivity of vanilla tensor-product attention (TPA) in small ViTs while preserving its KV-friendly factorization. In particular, we keep (R_K, R_V) unchanged so that the KV storage cost remains controlled, and introduce a lightweight nonlinearity to enrich the per-head representations with minimal additional parameters.

4.1. A rank-induced subspace constraint in vanilla TPA

Vanilla TPA constructs each per-token query slice $Q_t \in \mathbb{R}^{h\times d_h}$ as

$$Q_t = \frac{1}{R_Q} A_Q(x_t)^\top B_Q(x_t),$$

where $A_Q(x_t) \in \mathbb{R}^{R_Q\times h}$ and $B_Q(x_t) \in \mathbb{R}^{R_Q\times d_h}$ are contextual factors produced from the token representation x_t (see Sec. X for details). For a fixed head $i \in [h]$, define the per-head query vector $q_{t,i} = Q_t[i,:] \in \mathbb{R}^{d_h}$. Then

$$q_{t,i} = \frac{1}{R_Q}\sum_{r=1}^{R_Q} A_Q(x_t)[r,i]\; B_Q(x_t)[r,:] = \frac{1}{R_Q}\sum_{r=1}^{R_Q} \alpha_{t,i,r}\, b_{t,r},$$

where $b_{t,r} := B_Q(x_t)[r,:] \in \mathbb{R}^{d_h}$ and $\alpha_{t,i,r} := A_Q(x_t)[r,i]$. Hence, for each token t, $q_{t,i} \in \operatorname{span}\{b_{t,1},\dots,b_{t,R_Q}\}$, i.e., the per-head query vector lies in an at-most R_Q-dimensional subspace. This linear-span constraint provides a concrete explanation for why increasing R_Q is the main remaining knob for expressivity under a tight KV budget, yet it is parameter-inefficient (Sec. 3).

An identical derivation holds for K_t and V_t, implying that $k_{t,i} \in \operatorname{span}\{b^K_{t,r}\}_{r=1}^{R_K}$ and $v_{t,i} \in \operatorname{span}\{b^V_{t,r}\}_{r=1}^{R_V}$.

4.2. NonlinearTPA: breaking the linear-span constraint with a lightweight MLP

To enrich the representation beyond a fixed linear span, we apply a shared MLP $g(\cdot)$ to each per-head vector:

$$q'_{t,i} = g(q_{t,i}), \qquad g : \mathbb{R}^{d_h} \to \mathbb{R}^{d_h}.$$

In practice, we reshape the per-token tensor into (B, N, h, d_h) and apply the same $g(\cdot)$ to every length-d_h vector (across batches, tokens, and heads).

Single-hidden-layer MLP with width ratio. We instantiate $g(\cdot)$ as a two-layer feed-forward network with a single hidden layer and a GELU nonlinearity:

$$g(x) = W_2\,\mathrm{GELU}(W_1 x), \qquad W_1 \in \mathbb{R}^{m\times d_h},\ W_2 \in \mathbb{R}^{d_h\times m},$$

where the hidden width is controlled by a ratio hyperparameter ρ,

$$m = \rho\, d_h.$$

Unless otherwise stated, we use the same ρ for all layers and share the MLP parameters across all heads.

Preserving KV-friendly structure. Crucially, this nonlinearity does not change the factor dimensions nor the rank hyperparameters (R_Q, R_K, R_V). Therefore, the amount of stored KV factors (and thus the normalized KV storage cost defined in Sec. 3) remains unchanged compared to the corresponding vanilla TPA configuration; our method only adds a small number of extra parameters.

Parameter overhead. Ignoring biases, the added parameters of the shared MLP are

$$\Delta\#\mathrm{Params} \approx 2\, d_h\, m = 2\rho\, d_h^2,$$

which is independent of the sequence length and typically much smaller than increasing R_Q under a tight KV budget (Sec. 3).

4.3. MLP Placement Options

We treat the MLP insertion point as a design choice and define a small family of NonlinearTPA modules. —— We study several ways to insert the nonlinearity, differing only in which of $Q/K/V$ is transformed by an MLP. All options keep the same rank hyperparameters (R_Q, R_K, R_V) as the corresponding vanilla TPA baseline.

- **Q**: apply $g(\cdot)$ only to per-head query vectors.
- **K**: apply $g(\cdot)$ only to per-head key vectors.
- **V**: apply $g(\cdot)$ only to per-head value vectors.
- **KV**: apply separate MLPs to keys and values, i.e., $K' = g_K(K)$ and $V' = g_V(V)$.
- **KV_shared**: apply a single shared MLP to both keys and values, i.e., $K' = g(K)$ and $V' = g(V)$ with the same $g(\cdot)$.
- **QKV**: apply separate MLPs to queries, keys, and values, i.e., $Q' = g_Q(Q)$, $K' = g_K(K)$, and $V' = g_V(V)$.

We evaluate these placement options in Sec. 5.

4.4. Head-wise NonlinearTPA

Beyond the shared-MLP design, we consider a *head-wise* variant that uses an independent MLP per attention head. Let $x_{t,i} \in \mathbb{R}^{d_h}$ denote the per-head vector (query/key/value) of token t at head $i \in [h]$. We apply a head-specific MLP $g_i(\cdot)$:

$$x'_{t,i} = g_i(x_{t,i}), \qquad i \in [h].$$

In implementation, we slice along the head dimension and apply $g_i(\cdot)$ to each (B, N, d_h) block. This increases parameters by roughly a factor of h compared to a shared MLP, while keeping (R_Q, R_K, R_V) (and thus KV storage cost) unchanged. We evaluate the head-wise variant in Sec. 5.

5. Experiments

We evaluate NonlinearTPA on image classification using two standard benchmarks, CIFAR-10 and CIFAR-100. Our goal is to test whether a lightweight nonlinearity can consistently improve vanilla TPA under a fixed rank configuration and a nearly matched parameter budget across MLP placements. Unless stated otherwise, we report *best validation* top-1 accuracy, along with parameter counts.

5.1. Experimental Setup

Backbone We use a ViT-Tiny backbone from the `timm` library (`vit_tiny_patch16_224`) and replace its attention module with MHA, vanilla TPA, or our NonlinearTPA variants. In our setup, ViT-Tiny uses h=3 attention heads with per-head dimension d_h=64 (thus D=hd_h).

Datasets We consider CIFAR-10 and CIFAR-100. All methods are trained and validated using the same data processing and augmentation pipeline.

Baselines We compare against standard multi-head attention (MHA) and vanilla TPA. Unless otherwise stated, we fix $(R_q, R_k, R_v) = (16, 2, 2)$ for all TPA-based models, which yields the best validation accuracy among the KV-budgeted rank settings we tested and serves as a strong vanilla-TPA baseline for our ablations. All NonlinearTPA variants are evaluated on top of this same rank configuration, so the comparisons isolate the effect of the MLP design rather than changes in (R_q, R_k, R_v). Complementary rank trade-offs (e.g., lower-R_k/R_v settings) are analyzed in Sec. 3.

NonlinearTPA variants NonlinearTPA applies a single-hidden-layer MLP with GELU nonlinearity to per-head vectors, where the hidden width is $m = \rho d_h$. We evaluate multiple MLP placement options (e.g., Q, K, V, KV, KV_shared, and QKV) and a head-wise variant (Sec. 4.4).

Parameter-budget matching To isolate the effect of MLP placement, we tune the ratio ρ so that different NonlinearTPA placements incur nearly identical MLP parameter overhead (i.e., similar total parameter counts within each dataset).

Metrics We report best validation top-1 accuracy and the total number of parameters (in millions). We refer to Sec. 3 for the KV storage-cost proxy used to motivate KV-friendly designs.

Additional details Full training details—including the optimizer, learning-rate schedule, batch size, number of epochs, weight decay, and other hyperparameters—as well as training/validation curves and the evolution of validation accuracy across epochs are provided in the appendix.

Table 2. NonlinearTPA placement ablation on CIFAR-10/100 with TPA ranks $(R_q, R_k, R_v) = (16, 2, 2)$. The MLP hidden width is $m = \rho d_h$ (here d_h=64). Parameter counts differ slightly across datasets due to the classifier head (10 vs. 100 classes).

Method	MLP_on	ρ	Params10	Acc10	Params100	Acc100
MHA	–	–	5.526	**0.7452**	5.544	**0.4457**
TPA (baseline)	–	–	7.296	0.7050	7.313	0.3925
NonlinearTPA	KV	1.5	7.594	0.7257	7.611	0.4141
NonlinearTPA	KV_shared	3.0	7.594	0.7178	7.611	0.4123
NonlinearTPA	QKV	1.0	7.594	**0.7276**	7.611	**0.4240**
NonlinearTPA	Q	3.0	7.594	0.7215	7.611	0.4164
NonlinearTPA	K	3.0	7.594	0.7188	7.611	0.4070
NonlinearTPA	V	3.0	7.594	0.7103	7.611	0.4140

Table 3. CIFAR-100: token-wise vs. head-wise (HW) NonlinearTPA under matched total MLP parameter budget. All models use TPA ranks $(R_q, R_k, R_v) = (16, 2, 2)$.

Variant	MLP_on	ρ	Params(M)	Val Acc
Token-wise	KV	1.50	7.611	0.4141
Head-wise	KV	0.67	7.611	**0.4170**
Token-wise	KV_shared	3.00	7.611	0.4123
Head-wise	KV_shared	1.00	7.611	**0.4200**

5.2. Main Results and MLP Placement Ablation

Table 2 reports validation performance on CIFAR-10 and CIFAR-100 for MHA, vanilla TPA, and our NonlinearTPA variants under the same TPA rank configuration $(R_q, R_k, R_v) = (16, 2, 2)$. Across both datasets, introducing a lightweight nonlinearity consistently improves vanilla TPA. The best-performing placement, **QKV**, increases top-1 validation accuracy from 0.7050 to 0.7276 on CIFAR-10 ($+2.26$ points) and from 0.3925 to 0.4240 on CIFAR-100 ($+3.15$ points). These gains are achieved under a nearly matched parameter budget across placements by tuning the MLP width ratio ρ (Sec. 5.1).

Compared to MHA, vanilla TPA exhibits a clear accuracy gap in this small-model regime. NonlinearTPA reduces this gap substantially: on CIFAR-10, the gap shrinks from 0.0402 (MHA vs. TPA) to 0.0176 (MHA vs. NonlinearTPA-QKV), and on CIFAR-100, from 0.0532 to 0.0217. We also observe that *where* the MLP is applied matters. Applying the MLP to all of Q, K, V (QKV) performs best on both datasets, while applying it to a single component (Q/K/V) yields smaller improvements; in particular, V-only provides the weakest gains on CIFAR-10. Interestingly, using separate MLPs for K and V (KV) outperforms sharing a single MLP between them (KV_shared) under the same parameter budget, suggesting that decoupling key/value nonlinearities can be beneficial.

5.3. Head-wise vs. Token-wise Nonlinearity

We further compare the default token-wise MLP (shared across all heads) with a head-wise (HW) variant that uses an independent MLP per head. To ensure a fair comparison, we match the *total* MLP parameter budget between the two by adjusting the width ratio ρ (thus each head-wise MLP is narrower, but there are h of them). Table 3 shows that the head-wise design yields small but consistent gains in the two settings we tested: $0.4141 \rightarrow 0.4170$ for KV and $0.4123 \rightarrow 0.4200$ for KV_shared.

However, these improvements should be interpreted cautiously. ViT-Tiny has only $h{=}3$ heads, and our results are based on single runs, so run-to-run variance may partially explain the observed differences. Nevertheless, the head-wise formulation is an interesting direction: since capacity can be increased by scaling the per-head MLP ratio, it may offer a higher expressivity ceiling under larger models or more thorough tuning. We leave a more systematic study (e.g., multiple seeds and larger backbones) to future work.

6. Conclusion and Limitations

We studied the expressivity bottleneck of vanilla tensor-product attention (TPA) in small vision transformers and proposed *NonlinearTPA*, a lightweight modification that injects a simple GELU-MLP into factorized TPA while keeping the underlying rank configuration unchanged. Motivated by the rank-induced linear-span constraint in vanilla TPA (Sec. 3), our method aims to improve representation capacity without sacrificing TPA's KV-friendly structure.

Empirically, on ViT-Tiny with fixed ranks $(R_q, R_k, R_v) = (16, 2, 2)$, NonlinearTPA consistently improves vanilla TPA on both CIFAR-10 and CIFAR-100. Under a nearly matched MLP parameter budget across placements, applying the MLP to all of Q, K, V performs best, yielding a gain of $+2.26$ points on CIFAR-10 and $+3.15$ points on CIFAR-100 over the vanilla TPA baseline (Table 2). We also explored a head-wise variant that allocates separate MLPs per head; while it shows small gains in our preliminary tests, its effect appears sensitive in this small-h regime and warrants further validation (Table 3).

Limitations Our evaluation is limited to small-scale image classification (CIFAR-10/100) with a single backbone and a fixed rank setting, and most results are reported from single runs. A more comprehensive study on larger backbones, multiple seeds, additional rank configurations, and downstream tasks where KV memory is a primary bottleneck would help clarify the robustness and scaling behavior of NonlinearTPA.

Future Work

Reproducibility For all results reported in Tables 1–3, we plan to rerun experiments under a fully *frozen* codebase with fixed training scripts and hyperparameters, evaluate each setting with 3–5 random seeds, and report mean±std in addition to best validation accuracy. We will also release the code (and configuration files) to facilitate reproduction.

Broader evaluation We plan to extend the study beyond ViT-Tiny on CIFAR to larger ViT backbones and additional vision datasets, and evaluate whether the observed gains persist as h and d_h increase.

Statement of Contributions

This report is authored by a single person (Qiwu Wen). The author defined the problem setting, reviewed relevant literature, designed the methodology, implemented all models and training code, conducted all experiments, performed the analysis, prepared the presentation slides, and wrote the entire report. The project was conducted under the guidance of the course instructor, who provided high-level feedback on the experimental design and methodological choices.

We hereby state that all the work presented in this report is that of the author.

References

Ainslie, J., Lee-Thorp, J., De Jong, M., Zemlyanskiy, Y., Lebrón, F., and Sanghai, S. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

Deng, G., Liu, M., Wu, D., Li, Y., and Song, L. Enhancing low-rank adaptation with structured nonlinear transformations. *arXiv preprint arXiv:2509.21870*, 2025.

Dong, H., Zhu, W., Song, G., and Wang, L. Aurora: Breaking low-rank bottleneck of lora with nonlinear mapping. *arXiv preprint arXiv:2505.18738*, 2025.

Dosovitskiy, A. et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Liu, X., Peng, H., Zheng, N., Yang, Y., Hu, H., and Yuan, Y. Efficientvit: Memory efficient vision transformer with cascaded group attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14420–14430, 2023.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

Shazeer, N. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, W., Xie, E., Li, X., Fan, D.-P., Song, K., Liang, D., Lu, T., Luo, P., and Shao, L. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 568–578, 2021.

Zhang, P., Dai, X., Yang, J., Xiao, B., Yuan, L., Zhang, L., and Gao, J. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2998–3008, 2021.

Zhang, Y., Liu, Y., Yuan, H., Qin, Z., Yuan, Y., Gu, Q., and Yao, A. C.-C. Tensor product attention is all you need. *arXiv preprint arXiv:2501.06425*, 2025.

A. Appendix A:Training Details and Learning Curves

A.1. Training setup

All models are trained with the same overall recipe: ViT-Tiny backbone (`vit_tiny_patch16_224` in `timm`), AdamW optimizer (learning rate 3×10^{-4}, weight decay 0.05), batch size 128, and cross-entropy loss. We train for 30 epochs on CIFAR-10 and 20 epochs on CIFAR-100. For preprocessing, images are resized to 224×224, randomly horizontally flipped, converted to tensor, and normalized with mean $(0.5, 0.5, 0.5)$ and std $(0.5, 0.5, 0.5)$. We do not use AMP, Mixup/CutMix, RandAugment, or label smoothing.

A.2. Evaluation protocol and reproducibility note

We report *best* validation accuracy within each run (maximum over epochs), consistent with the main tables.

Reproducibility note. The main tables summarize results collected from multiple exploratory runs during development. Since these runs were produced while iterating on the codebase and were not all executed with a single frozen training script and a fixed random seed, exact values may vary upon rerunning. Nevertheless, we expect the qualitative trends reported in the paper (e.g., consistently improving over vanilla TPA under a comparable MLP budget) to remain stable.

A.3. Learning curves

Figures 1–2 show representative validation accuracy curves on CIFAR-10 and CIFAR-100, respectively. These curves are included to illustrate training dynamics and to verify that the reported improvements are not driven by isolated spikes in a single epoch. We note that under our short training budget, the curves may not be fully converged; however, all methods are compared under the same training schedule.

Why we omit head-wise learning curves We do not include learning curves for the head-wise (HW) variants. In our small ViT-Tiny setting (h=3 heads), the observed HW gains are minor and not yet statistically conclusive under the current evaluation budget. We therefore present only the summarized comparison in the main paper and leave a more thorough study (e.g., multi-seed evaluation and larger-head models) to future work.

Camera-ready plan For a camera-ready submission, we will re-run the main comparisons using a frozen codebase with fixed random seeds (and, where appropriate, multi-seed reporting) to ensure exact reproducibility of the numbers reported in the tables and figures.

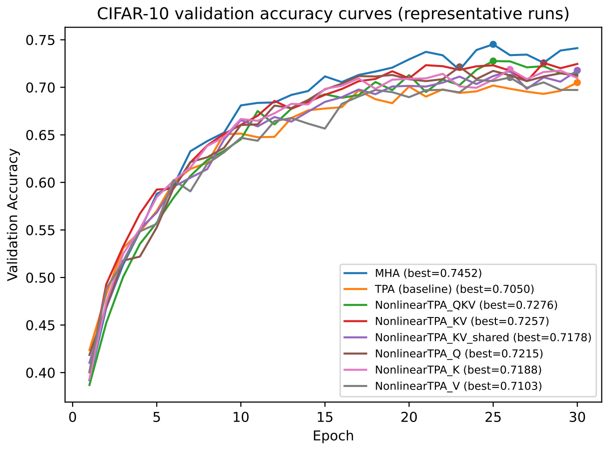

Figure 1. CIFAR-10 validation accuracy curves from representative runs.

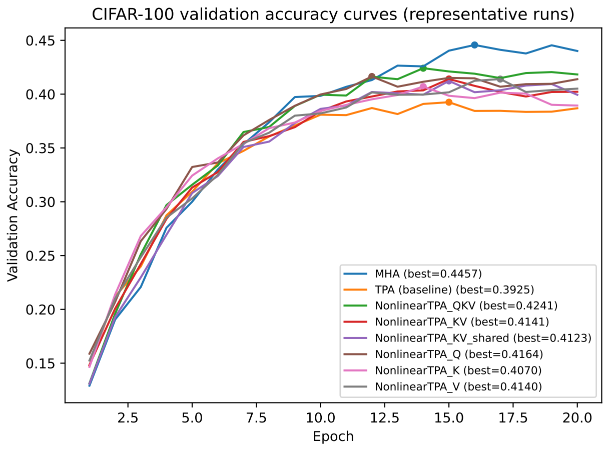

Figure 2. CIFAR-100 validation accuracy curves from representative runs.