# TWO-PHASE CACHING AND ACTIVATION FOR INPUT-EFFICIENT FINE-TUNING OF VISION TRANSFORMERS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Following the success of the ViT, transformers have been widely adopted for image recognition, multimodal understanding, and visual generation, increasing the need for efficient fine-tuning. Although parameter-efficient fine-tuning (PEFT) reduces trainable parameters, it can still require back-propagation through the full token sequence, retaining substantial activation and computation costs. Token pruning and merging address this overhead by shortening the sequence, but also modify the context available during forward propagation. Building on selective token back-propagation, we formulate input-efficient fine-tuning (IEFT) for vision transformers to reduce token-level backward computation while retaining full-sequence forward context. To enable token selection informed by the completed forward pass, we propose a two-phase caching-activation (TCA) strategy that first caches global keys and values during a gradient-free full forward pass, then recomputes selected tokens against this cached context. With fixed parameters and consistent deterministic computation, this replay reproduces the corresponding full-sequence forward outputs while restricting gradient propagation to selected token paths. Experiments across 13 recognition tasks demonstrate over 60% peak-memory reduction and approximately 45% training-time reduction with minimal accuracy degradation on ViT fine-tuning. Combining TCA with PEFT yields additional efficiency gains. Extensions to Qwen3-VL-8B, FLUX.2-4B, and Wan2.2-14B cover multimodal understanding and visual generation, yielding $3.4\times$ average micro-batch capacity for the first two and approximately $2\times$ video generation training throughput on the same hardware. **The code is available in the supplementary materials and will be publicly released.**

## 1 INTRODUCTION

Following the success of ViT in image recognition Dosovitskiy et al. (2021); Liu et al. (2021); Mehta & Rastegari (2022), transformer architectures have been widely adopted in multimodal large language models (MLLMs) Zhu et al. (2025); Bai et al. (2025) and large-scale visual generation models Peebles & Xie (2023); Wu et al. (2025); Labs (2025); Wan et al. (2025). Despite their impressive performance, these vision transformers usually involve high computational complexity and an ever-increasing number of parameters. As a result, fine-tuning such models often requires substantial computation resources, which limits their practicality in application scenarios and edge devices. To alleviate this, parameter-efficient fine-tuning (PEFT) has emerged as the mainstream paradigm, aiming to reduce the training cost by freezing most pre-trained weights and only updating a minimal set of parameters.

Existing PEFT methods for vision transformers can be broadly viewed as different ways of introducing lightweight adaptation into a frozen backbone. One branch augments internal computation with trainable modules, including adapters and mixture-of-experts (MoE) blocks Chen et al. (2022); Cai et al. (2025), as well as vision-oriented LoRA variants Jie & Deng (2023); Khanna et al. (2025) that approximate dense updates through low-rank parameterization in self-attention. Another branch intervenes at the input level, where visual prompt tuning (VPT) Ma et al. (2025); Jia et al. (2022) prepends learnable tokens to steer the pre-trained model toward downstream tasks. Finally, selective fine-tuning and feature modulation techniques Sung et al. (2022); Lian et al. (2022) continue to offer extremely lightweight solutions by updating or re-using only a tiny subset of the original parameters.



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.

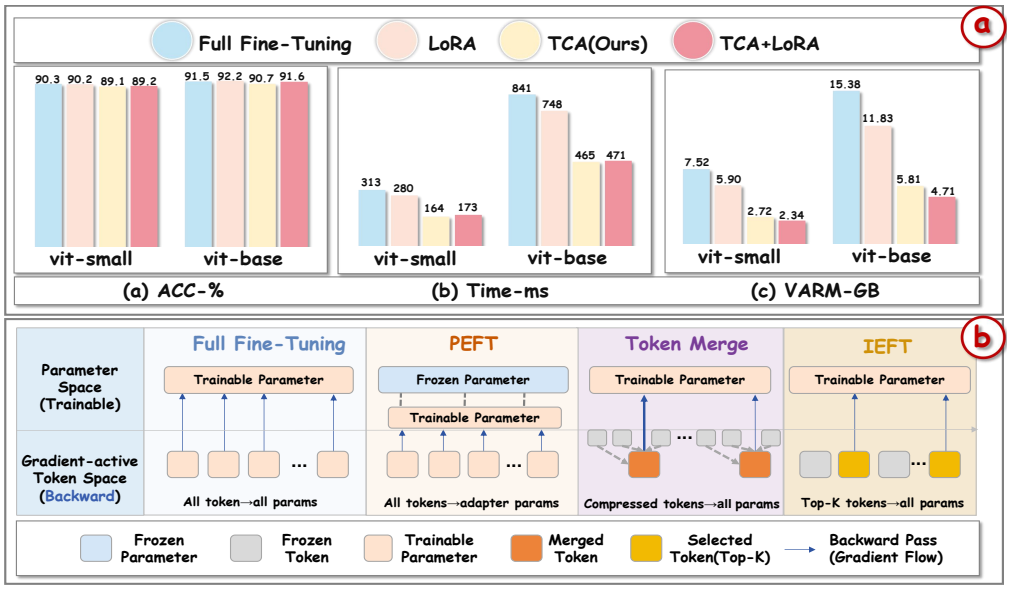



Figure 1: As illustrated in (a), on six datasets, applying our TCA to either full fine-tuning or PEFT paradigm yields only negligible accuracy degradation across different models. Meanwhile, TCA achieves substantial reductions in fine-tuning cost. As illustrated in (b), IEFT restricts back-propagation to selected tokens while retaining the full token sequence during the forward pass.

However, while PEFT reduces parameter-related optimization costs, substantial token-level activation and backward computation costs remain. To route gradients to trainable modules, the network still requires intermediate activations over the full token sequence. Forward token pruning and merging methods attempt to mitigate this by pruning or merging tokens during the forward pass Rao et al. (2021); Li et al. (2025); Liang et al. (2022). Nevertheless, these approaches alter the forward context available to the remaining tokens, potentially affecting adaptation performance, especially under aggressive token reduction. A natural alternative is selective token back-propagation, which targets backward costs while preserving full-sequence forward context.

Prior work has explored selective back-propagation in high-resolution image recognition through the multiple-instance learning architecture Qin et al. (2025), and in text-only language-model fine-tuning through TokenTune Simoulin et al. (2024) and TokenSeek Zeng et al. (2026). Building on these advances, as shown in Fig. 1 (b), we formulate input-efficient fine-tuning (IEFT) for vision transformers, retaining full-sequence forward context while restricting back-propagation to selected tokens. Furthermore, to enable token selection informed by the completed forward pass, we propose a two-phase caching-activation (TCA) strategy. Specifically, TCA decouples the training pipeline into a gradient-free caching phase and a gradient-aware activation phase. During the initial caching phase, the framework executes a gradient-free forward pass to cache the global key and value (KV) representations for all tokens layer-by-layer. Subsequently, in the activation phase, we first identify a small subset of active tokens based on the attention distribution. By leveraging the pre-cached global KV information, the model performs layer-wise computation exclusively for this active subset. This replay reproduces the selected-token forward outputs under the conditions stated above, while restricting gradient propagation to selected token paths. As illustrated in Fig. 1 (a), TCA fundamentally reduces the fine-tuning cost with negligible accuracy degradation. Extensive experiments across 13 recognition tasks demonstrate over 60% peak-memory reduction and approximately 45% training-time reduction with minimal accuracy degradation on ViTs, while separate LoRA experiments verify TCA's compatibility with PEFT. We further extends TCA to multimodal understanding with Qwen3-VL-8B Bai et al. (2025) and image generation with FLUX.2-4B Labs (2025), supporting $3.4\times$ the micro-batch size of full-token LoRA on average, while doubling video-training throughput with Wan2.2-14B Wan et al. (2025) on matched hardware.

Our contributions are three-fold: 1) We formulate input-efficient fine-tuning (IEFT) for vision transformers, building on selective back-propagation to reduce token-dependent training costs while retaining full-sequence forward context. 2) We propose a two-phase caching–activation (TCA) strategy that separates gradient-free context caching from gradient-aware token recomputation, en-

abling active-token selection informed by the completed full-sequence forward pass. 3) Experiments across 13 recognition tasks demonstrate over 60% peak-memory reduction and approximately 45% training-time reduction with minimal accuracy degradation on ViTs. Further experiments verify compatibility with LoRA and applicability to multimodal understanding and visual generation.

## 2 Related Work

### 2.1 Parameter-Efficient Fine-Tuning in Vision Transformers

To adapt pre-trained vision transformers to downstream tasks with lower training cost, parameter-efficient fine-tuning (PEFT) has become the dominant paradigm. Its central idea is to freeze the pre-trained backbone and update only a lightweight set of task-specific parameters. PEFT methods mainly differ in how they parameterize task-specific adaptation. AdaptFormer, Convpass, and VL-Adapter introduce lightweight trainable modules Chen et al. (2022); Jie et al. (2024a); Sung et al. (2022), whereas FacT factorizes weight increments Jie & Deng (2023) and SSF learns feature-wise scaling and shifting Lian et al. (2022). Visual prompt tuning learns auxiliary prompts Jia et al. (2022), with extensions incorporating label correlations, image-specific information, and multi-scale features Ma et al. (2025); Liu et al. (2023b); Hossain et al. (2024). Other studies investigate automatic adaptation-module design Zhang et al. (2024), expert-based adaptation for tracking Cai et al. (2025), and parameter-efficient extended pre-training under domain shifts Khanna et al. (2025). Despite these differences, compact parameterizations do not necessarily remove the activations required to propagate gradients through frozen layers. IEFT addresses this complementary dimension by restricting gradient-active tokens rather than further reducing trainable parameters.

### 2.2 Selective Token Computation

Visual token computation can be reduced through pruning Rao et al. (2021); Liang et al. (2022) or merging Bolya et al. (2023); Zeng et al. (2022), with subsequent work exploring adaptive and training-aware compression Chen et al. (2023); Jie et al. (2024b); Li et al. (2025). ZeroTPrune Wang et al. (2024) further exploits pre-trained attention structures for zero-shot token pruning. These methods shorten the forward sequence but alter the context available to subsequent layers. Differently, selective back-propagation restricts gradient computation while retaining contextual information. In high-resolution image recognition, DBPS Qin et al. (2025) combines gradient-enabled salient patches with gradient-free contextual patches. For language models, TokenTune Simoulin et al. (2024) retains full-sequence forward context while restricting back-propagation to randomly selected tokens. TokenSeek Zeng et al. (2026) instead uses precomputed attention- and gradient-based importance scores that remain fixed during fine-tuning. Building on these studies, we formulate IEFT for vision transformers and propose TCA, which separates full-sequence KV caching from selected-token recomputation, enabling selection informed by the completed current forward pass. During replay, selected tokens retain access to unselected tokens through their cached representations, without recomputing the unselected queries and feed-forward operations.

## 3 Method

### 3.1 Efficiency Analysis of Full Fine-Tuning

To analyze the back-propagation cost of vision transformer, we establish a complexity model for full fine-tuning under automatic differentiation. We focus on three variables: the batch size $B$, sequence length $N$, and channel hidden dimension $D$, while omitting network depth and constant factors for asymptotic clarity. During the back-propagation, we group the dominant costs into three gradient terms. For parameterized layers, deriving the weight gradients $\nabla \boldsymbol{W}$ and the token activation gradients $\nabla \boldsymbol{X}$ requires two matrix multiplications of the same asymptotic order. We denote their respective computational costs as $\mathcal{T}(\nabla \boldsymbol{W})$ and $\mathcal{T}(\nabla \boldsymbol{X})$, with $\mathcal{T}(\nabla \boldsymbol{W}) = \mathcal{T}(\nabla \boldsymbol{X}) = \mathcal{O}(B \cdot N \cdot D^2)$. In addition, routing token activation gradients through the attention probability matrix $\boldsymbol{A}$ incurs an extra back-propagation cost associated with $\nabla \boldsymbol{A}$, which scales as $\mathcal{T}(\nabla \boldsymbol{A}) = \mathcal{O}(B \cdot N^2 \cdot D)$.

For memory analysis, we consider the additional memory required for training, excluding the resident backbone weights shared by all methods. This additional memory consists of static training

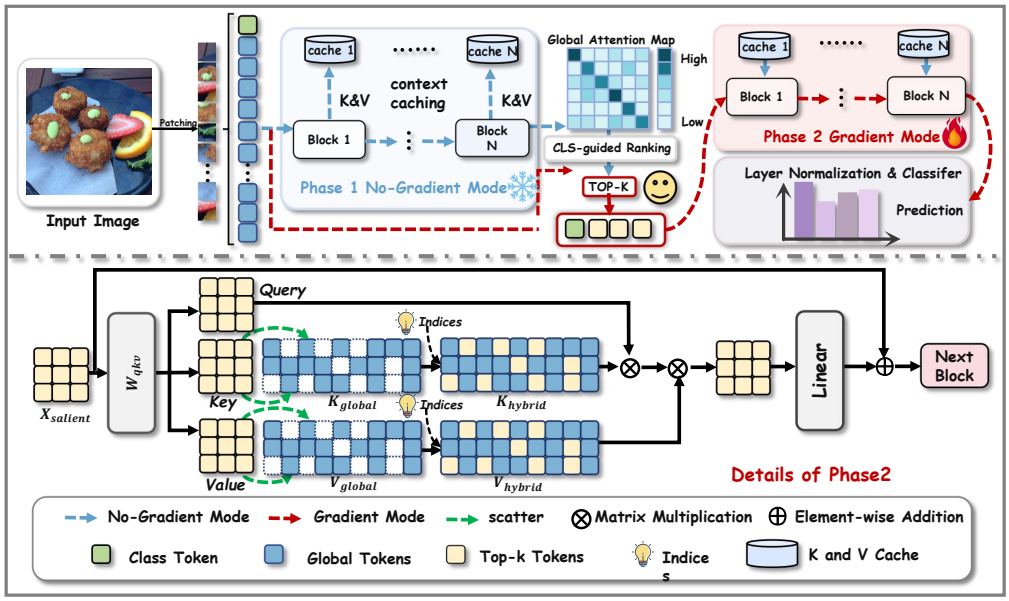



Figure 2: Overview of the two-phase caching-activation (TCA) strategy. Phase 1 performs a gradient-free full forward pass and caches layer-wise keys and values. Phase 2 replays selected tokens using the cached full-sequence context, restricting gradient propagation to selected paths.

states and dynamic activations. First, the static memory stores the gradients of weights $\nabla \boldsymbol{W}$ and optimizer states, which we denote by $\mathcal{M}(\boldsymbol{W}) = \mathcal{O}(D^2)$ and treat as batch-invariant. Second, the dynamic memory stores the batch-proportional token activations required for back-propagation. Beyond the linear token states $\mathcal{M}(\boldsymbol{X}) = \mathcal{O}(B \cdot N \cdot D)$, the dominant term comes from the dense attention map, whose memory scales as $\mathcal{M}(\boldsymbol{A}) = \mathcal{O}(B \cdot N^2)$.

### 3.2 Efficiency Analysis of PEFT

PEFT reduces training cost by freezing the pre-trained dense weights $\boldsymbol{W}$ and introducing a small set of trainable adapters with rank $r \ll D$. By freezing the dense backbone, PEFT largely removes the computation and storage associated with the full weight gradients $\nabla \boldsymbol{W}$. Consequently, the trainable weight-gradient cost is reduced to $\mathcal{T}(\nabla \boldsymbol{W}_p) = \mathcal{O}(B \cdot N \cdot r \cdot D)$. And the static parameter-related memory also drops to $\mathcal{M}(\boldsymbol{W}_p) = \mathcal{O}(r \cdot D)$. However, this reduction mainly affects parameter-related overhead rather than token-dependent activations. As a result, the dominant back-propagation costs associated with $\mathcal{T}(\nabla \boldsymbol{X})$ and $\mathcal{T}(\nabla \boldsymbol{A})$ remain of the same asymptotic order as in full fine-tuning. Likewise, the dominant batch-proportional activation memory, including the saved token states $\mathcal{M}(\boldsymbol{X}) = \mathcal{O}(B \cdot N \cdot D)$ and attention maps $\mathcal{M}(\boldsymbol{A}) = \mathcal{O}(B \cdot N^2)$, remains asymptotically unchanged. This observation indicates that PEFT alleviates the static parameter overhead but does not resolve the dynamic token activation bottleneck.

### 3.3 Input-Efficient Fine-Tuning via TCA

The analysis in Secs. 3.1 and 3.2 reveals a fundamental bottleneck: when the full-sequence context is preserved during the forward pass and all tokens remain active during back-propagation, the costs associated with $\nabla \boldsymbol{X}$ and $\nabla \boldsymbol{A}$ remain tied to the full token dimension $N$. To address this issue, we formalize a training paradigm termed input-efficient fine-tuning (IEFT). Analogous to how PEFT restricts the trainable parameter space, IEFT restricts the gradient-active token space while retaining the full-sequence context for forward computation. Specifically, IEFT partitions the full input sequence $\boldsymbol{X}$ into a compact gradient-active subset $\boldsymbol{X}_a$ and a gradient-detached context subset $\boldsymbol{X}_d$:

$$\boldsymbol{X} = \boldsymbol{X}_a \cup \boldsymbol{X}_d, \qquad \nabla \boldsymbol{X} = \nabla \boldsymbol{X}_a \cup \nabla \boldsymbol{X}_d = \nabla \boldsymbol{X}_a, \qquad |\boldsymbol{X}_a| = K < N. \tag{1}$$

In Eq. 1, $\nabla \boldsymbol{X}$ denotes the collection of token-level gradient terms computed along the gradient-active paths of IEFT. The detached subset contributes no entries to these collections. Thus, the

forward pass retains the context of all $N$ tokens, whereas token-level back-propagation is restricted to the $K$ active tokens. To select gradient-active tokens using full-sequence forward information without retaining the full-token computation graph, we introduce a two-phase caching-activation (TCA) strategy, comprising a gradient-free caching phase and a gradient-aware activation phase.

#### 3.3.1 GRADIENT-FREE CACHING PHASE.

In the first gradient-free caching phase, TCA executes a complete forward pass to capture and cache the full-sequence context. This phase is performed under a gradient-free setting (e.g., `torch.no_grad()`), without constructing an autograd graph. For a standard self-attention transformer block, the forward update of an individual token $\boldsymbol{x}_i$ at layer $l$ can be expressed as a function of its own local state and the full-sequence KV representations:

$$\boldsymbol{x}_i^{l+1} = \mathcal{F}^l\Big(\boldsymbol{x}_i^l,\ \mathcal{K}^l(\boldsymbol{X}^l),\ \mathcal{V}^l(\boldsymbol{X}^l)\Big) \tag{2}$$

*This structural property implies that, once the full-sequence KV representations are available, selected token outputs can be recomputed without constructing an autograd graph for all tokens.* Accordingly, TCA caches the gradient-free representations $\mathcal{K}^l(\boldsymbol{X}^l)$ and $\mathcal{V}^l(\boldsymbol{X}^l)$ at each layer during this phase. The caching pass also provides information for active-token selection, whose criterion depends on the task and model architecture. For classification models equipped with a [CLS] token, we use its final-layer attention distribution $\boldsymbol{A}_{cls}^L \in \mathbb{R}^N$ as a task-conditioned proxy for token importance.

$$\mathcal{I}_a = \underset{i\in\{1,\dots,N\}}{\arg\text{top-}K}\ \boldsymbol{A}_{cls,i}^L, \quad \boldsymbol{X}_a = \{\boldsymbol{x}_i \mid i \in \mathcal{I}_a\} \tag{3}$$

As no computational graph is materialized, this phase locates the task-relevant regions and caches the global visual context without introducing train-time token activation gradients.

#### 3.3.2 GRADIENT-AWARE ACTIVATION PHASE.

In the subsequent gradient-aware activation phase, TCA constructs the token-level backbone computation graph only for the selected active subset $\boldsymbol{X}_a$, while retaining the cached full-sequence context. At each layer $l$, TCA recomputes the queries, keys, and values of the active tokens. The recomputed keys and values replace the corresponding entries in the gradient-free caches, allowing gradients to propagate through the active KV projections while keeping the remaining entries detached. For clarity, attention is written in single-head notation:

$$\boldsymbol{A}_a^l = \text{Softmax}\left(\frac{\mathcal{Q}^l(\boldsymbol{X}_a^l)\left(\mathcal{K}^l(\boldsymbol{X}^l) \cup \mathcal{K}^l(\boldsymbol{X}_a^l)\right)^\top}{\sqrt{D}}\right) \in \mathbb{R}^{B\times K\times N}. \tag{4}$$

The active-token update is then given by:

$$\boldsymbol{X}_a^{l+1} = \mathcal{F}^l\Big(\boldsymbol{X}_a^l, \mathcal{K}^l(\boldsymbol{X}^l) \cup \mathcal{K}^l(\boldsymbol{X}_a^l), \mathcal{V}^l(\boldsymbol{X}^l) \cup \mathcal{V}^l(\boldsymbol{X}_a^l)\Big) = \{\boldsymbol{x}_i^{l+1} \mid i \in \mathcal{I}_a\}. \tag{5}$$

Here, $\cup$ denotes index-preserving replacement of cached entries by their recomputed active-token counterparts, rather than concatenation or ordinary set union. The final equality in Eq. 5 states that replay reproduces the corresponding token outputs of the full-sequence forward pass. This holds when both phases use the same parameters and deterministic operations, with fixed active indices across layers and unchanged positional information and attention masks.

### 3.4 EFFICIENCY ANALYSIS OF TCA

Revisiting Secs. 3.1 and 3.2, unlike PEFT that mainly targets parameter-related training costs, TCA reduces token-dependent costs during back-propagation. During the activation phase, graph construction is restricted to $\boldsymbol{X}_a$, while full-sequence KV representations remain available for backward computation. The resulting token-state and explicitly materialized attention storage scales as:

$$\mathcal{M}(\boldsymbol{X}) + \mathcal{M}(\boldsymbol{A}_a) = \mathcal{O}(BND) + \mathcal{O}(BKN). \tag{6}$$

Here, $\mathcal{M}(\boldsymbol{X})$ includes the full-sequence KV representations and saved active-token states. Likewise, the leading token-dependent backward cost becomes:

$$\mathcal{T}(\nabla \boldsymbol{X}_a) + \mathcal{T}(\nabla \boldsymbol{A}_a) = \mathcal{O}(B\cdot K\cdot D^2) + \mathcal{O}(B\cdot K\cdot N\cdot D) = \mathcal{O}\big(B\cdot K\cdot D\cdot (D+N)\big). \tag{7}$$

In the activation phase, the additional active-token forward recomputation incurs:

$$\mathcal{T}_{extra} = \mathcal{O}\big(B \cdot K \cdot D \cdot (D + N)\big). \tag{8}$$

Meanwhile, the corresponding saving in token-dependent backward computation scales as:

$$\mathcal{T}_{save} = \mathcal{O}\big(B \cdot (N - K) \cdot D \cdot (D + N)\big). \tag{9}$$

Therefore, the net saving in these dominant computations is:

$$\Delta\mathcal{T} = \mathcal{T}_{save} - \mathcal{T}_{extra}. \tag{10}$$

Under linear token-wise scaling, if backward computation per token is no cheaper than forward replay, $K < N/2$ suffices for $\Delta\mathcal{T} > 0$. Realized speedups additionally depend on selection, cache management, and hardware efficiency. For memory, full-sequence KV storage contributes $\mathcal{M}\big(\mathcal{K}(\boldsymbol{X}), \mathcal{V}(\boldsymbol{X})\big) = \mathcal{O}(B \cdot N \cdot D)$, with depth suppressed as above. During activation, consumed KV caches are released layer by layer as replay proceeds. This progressively replaces caches with states retained for backward computation, limiting duplicate KV storage.

### 3.5 Efficiency Analysis of TCA with PEFT

TCA restricts the gradient-active token dimension, whereas low-rank PEFT reduces trainable parameters through a rank $r \ll D$. The two approaches therefore operate on complementary dimensions and can be combined naturally. Following the same depth-suppressed accounting and excluding model weights, the leading storage terms during activation comprise parameter gradients and optimizer states, full-sequence KV representations, and saved active-token states and attention maps:

$$\begin{aligned} \mathcal{M}_C &= \mathcal{M}(\boldsymbol{W}_p) + \mathcal{M}(\boldsymbol{X}) + \mathcal{M}(\boldsymbol{A}_a) \\ &= \mathcal{O}(r \cdot D) + \mathcal{O}(BND + BKN). \end{aligned} \tag{11}$$

Correspondingly, the leading backward computations within the adapted transformer blocks becomes:

$$\begin{aligned} \mathcal{T}_C &= \mathcal{T}(\nabla \boldsymbol{W}_p) + \mathcal{T}(\nabla \boldsymbol{X}_a) + \mathcal{T}(\nabla \boldsymbol{A}_a) \\ &= \mathcal{O}\big(B \cdot K \cdot D \cdot (r + D + N)\big). \end{aligned} \tag{12}$$

The combined scheme thus reduces parameter-related training storage and token-dependent backward costs, while retaining full-sequence context.

## 4 Experiment

### 4.1 Datasets and Evaluation Metrics

We validate the effectiveness of TCA across 13 downstream classification datasets. Specifically, we first evaluate TCA on general image classification tasks using four datasets: CIFAR-100 Krizhevsky et al. (2009), Food-101 Bossard et al. (2014), SVHN Netzer et al. (2011), and FI You et al. (2016). For domain-specific scenarios, we conduct experiments on CQU-BPDD Tang et al. (2021), DDR Li et al. (2019), NWPU Cheng et al. (2017), and ArtBench-10 Liao et al. (2022), corresponding to the pavement, medical, remote sensing, and art domains respectively. Finally, we test TCA on five classic fine-grained classification datasets: CUB-200 Wah et al. (2011), Stanford Cars Krause et al. (2013), NABirds Van Horn et al. (2015), Stanford Dogs Khosla et al. (2011), and Oxford Flowers Nilsback & Zisserman (2008). We report classification accuracy (%) and evaluate efficiency using peak GPU memory usage (VRAM, GB) and training-step time (Time, ms). Efficiency measurements for ViT classification experiments use a batch size of 128.

### 4.2 Implementation Details

We evaluate ViT-S/16, ViT-B/16, and ViT-L/16, replacing the pretrained classification head with a task-specific linear classifier. For a $224 \times 224$ input, the sequence contains 196 image patches and one classification token. Unless otherwise specified, TCA uses the classification token and 32 selected image tokens. We use AdamW Loshchilov & Hutter (2019) with a weight decay of 0.01. The initial learning rate is $5 \times 10^{-5}$ by default, $10^{-4}$ under PEFT settings, and $10^{-3}$ under

Table 1: Classification accuracy (%) on 13 downstream benchmarks and fine-tuning costs for three ViT variants. Linear denotes training only the linear classifier with the ViT backbone frozen.

| Model | Method | General | | | | | Domain-Specific | | | | | Fine-Grained | | | | | | Fine-Tuning Cost | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | C100 | Food | SVHN | FI | Avg. | BPDD | DDR | NWPU | Art | Avg. | CUB | Cars | NAB | Dogs | Flowers | Avg. | VRAM | Time |
| ViT-S/16 | Full | 90.3 | 89.3 | 97.7 | 72.2 | 87.4 | 79.9 | 72.4 | 93.1 | 71.2 | 79.2 | 83.5 | 84.5 | 79.7 | 83.5 | 98.1 | 85.9 | 7.5G | 313ms |
| | TCA | 89.1 | 88.1 | 97.5 | 72.0 | 86.7 | 79.7 | 72.7 | 92.1 | 69.7 | 78.6 | 83.4 | 84.1 | 78.8 | 82.2 | 97.4 | 85.2 | 2.7G | 164ms |
| | Linear | 81.9 | 78.7 | 44.7 | 65.8 | 67.6 | 59.7 | 65.3 | 81.8 | 56.1 | 65.7 | 78.6 | 40.9 | 68.3 | 78.9 | 95.8 | 72.5 | 0.6G | 84ms |
| ViT-B/16 | Full | 91.5 | 89.6 | 97.7 | 73.3 | 88.0 | 80.6 | 74.0 | 93.9 | 71.7 | 80.1 | 83.6 | 86.1 | 79.7 | 84.9 | 98.3 | 86.5 | 15.4G | 841ms |
| | TCA | 90.7 | 89.1 | 97.6 | 72.7 | 87.5 | 80.5 | 73.5 | 93.3 | 70.3 | 79.4 | 83.6 | 85.2 | 78.5 | 84.9 | 98.3 | 86.1 | 5.8G | 465ms |
| | Linear | 84.4 | 81.9 | 55.8 | 67.9 | 72.5 | 64.3 | 63.5 | 83.5 | 58.8 | 67.5 | 80.9 | 53.3 | 73.0 | 86.5 | 96.8 | 78.1 | 1.3G | 278ms |
| ViT-L/16 | Full | 93.0 | 92.4 | 98.1 | 76.4 | 90.0 | 74.0 | 75.8 | 95.2 | 73.5 | 79.6 | 89.5 | 90.1 | 87.5 | 90.6 | 99.5 | 91.4 | 42.5G | 2764ms |
| | TCA | 92.5 | 91.3 | 97.7 | 75.7 | 89.3 | 72.9 | 74.9 | 95.0 | 72.2 | 78.7 | 89.0 | 88.9 | 86.0 | 90.9 | 99.4 | 90.8 | 16.5G | 1501ms |
| | Linear | 89.5 | 90.0 | 67.5 | 72.4 | 79.9 | 68.9 | 69.3 | 91.1 | 65.9 | 73.8 | 87.6 | 67.4 | 82.2 | 91.5 | 99.0 | 85.5 | 2.4G | 1014ms |

Table 2: Combinations of IEFT (TCA) and PEFT (LORA) on six datasets.

| Model | Method | CIFAR-100 | FI | BPDD | DDR | Dogs | Flowers | Avg | VRAM | Time |
|---|---|---|---|---|---|---|---|---|---|---|
| ViT-S/16 | LoRA | 90.0 | 71.3 | 77.6 | 72.3 | 86.8 | 98.8 | 82.8 | 5.9G | 280ms |
| | + TCA | 89.2 | 70.5 | 77.2 | 72.5 | 86.2 | 98.4 | 82.3 | 2.3G | 173ms |
| ViT-B/16 | LoRA | 92.2 | 72.4 | 79.8 | 72.9 | 90.1 | 98.8 | 84.4 | 11.8G | 748ms |
| | + TCA | 91.6 | 72.2 | 80.1 | 73.5 | 89.6 | 98.7 | 84.3 | 4.7G | 471ms |

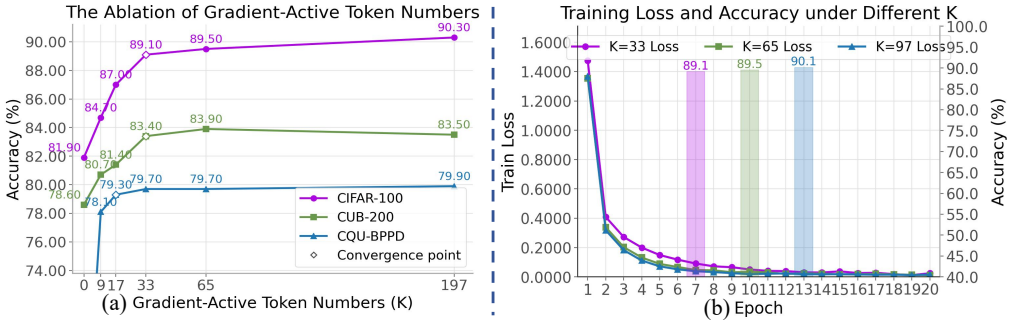



Figure 3: Ablation of the gradient-active token budget $K$ on ViT-S/16: (a) classification accuracy on three datasets; (b) training loss and test accuracy on CIFAR-100.

Linear settings. We measure runtime using `torch.cuda.Event` and GPU memory usage using `torch.cuda.memory_stats`. Total VRAM includes resident model parameters, gradients, optimizer states, and temporary tensors allocated during training. These experiments are implemented in PyTorch Paszke et al. (2019) and conducted on NVIDIA RTX 3090 GPUs. Further implementation details are provided in the Appendix.

### 4.3 Quantitative Results

**TCA reduces training costs with small accuracy losses.** Tab. 1 compares TCA with full fine-tuning and linear probing across 13 datasets and three ViT variants. For each model, the average accuracy drop relative to full fine-tuning remains below one percentage point within each of the three task categories. Meanwhile, peak memory usage decreases by approximately 61–64%, and training-step time decreases by 45–48%, demonstrating a favorable accuracy-efficiency trade-off. TCA also achieves substantially higher average accuracy than linear probing across all categories.

**TCA complements PEFT.** Tab. 2 evaluates TCA combined with LoRA on six datasets. Compared with LoRA alone, the combined method reduces average accuracy by only 0.5 and 0.1 percentage points on ViT-S/16 and ViT-B/16, respectively. It further reduces peak memory by approximately 60–61% and training-step time by 37–38%, supporting the complementarity of parameter-efficient adaptation and selective token back-propagation.

### 4.4 Ablation Study

**Effective adaptation does not require gradients through all image tokens.** Fig. 3(a) shows that back-propagating through a small token subset already recovers much of the accuracy of full fine-

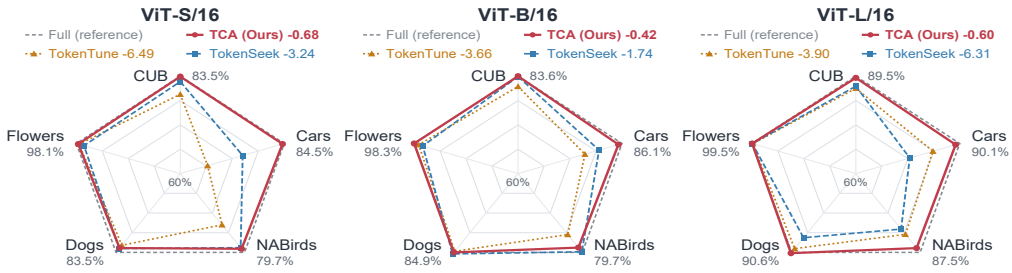



Figure 4: Comparison of TCA, TokenTune, and TokenSeek on five fine-grained tasks.

Table 3: Comparison and combination of TCA and token merging/pruning methods.

| Dataset | ViT-S/16 | | | | | ViT-B/16 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | TOME | MCTF | PM-ViT | TRAM | TCA+TRAM | TOME | MCTF | PM-ViT | TRAM | TCA+TRAM |
| CIFAR-100 | 77.9 | 65.4 | 55.7 | 63.4 | 85.7 | 79.8 | 83.9 | 77.5 | 69.9 | 87.6 |
| Food101 | 78.0 | 65.3 | 56.3 | 68.6 | 85.3 | 78.7 | 81.1 | 72.1 | 70.8 | 86.6 |
| Train/Test VRAM | 2.7 / 0.8 | 2.8 / 1.1 | 2.7 / 0.5 | 2.7 / 0.9 | 2.7 / 0.9 | 5.8 / 2.2 | 6.6 / 2.8 | 5.8 / 0.9 | 5.8 / 2.4 | 5.8 / 1.9 |

Table 4: Scaling experiments on Qwen3-VL-8B with LoRA. Base denotes the unfine-tuned model. TCA activates 30% of tokens. VRAM is summed over four A40 GPUs and E-VRAM excludes resident weights. Training and memory measurements use micro-batch 1 per GPU and 16-step accumulation. BS denotes the measured single-A40 micro-batch limit. Time estimates GPU-hours per epoch summed over 4 A40 GPUs, using each method's listed BS.

| Dataset | ACC (%) | | | VRAM (GB) | | Time (GPU-h) | | E-VRAM (GB) | | BS | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Base | LoRA | TCA | LoRA | TCA | LoRA | TCA | LoRA | TCA | LoRA | TCA |
| ScienceQA | 55.1 | 98.2 | 97.3 | 129.9 | 89.8 | 1.87 | 1.26 | 63.2 | 23.2 | 1 | 5 |
| OK-VQA | 56.9 | 67.2 | 67.3 | 110.8 | 83.4 | 2.39 | 1.71 | 44.2 | 16.8 | 2 | 7 |
| VSR | 85.1 | 89.4 | 89.3 | 110.9 | 83.3 | 2.01 | 1.43 | 44.2 | 16.7 | 2 | 7 |
| A-OKVQA | 88.6 | 89.3 | 89.8 | 105.7 | 82.5 | 4.49 | 3.29 | 37.7 | 14.6 | 2 | 7 |
| ChartQA | 83.4 | 85.5 | 85.3 | 104.9 | 82.8 | 7.27 | 5.09 | 37.0 | 14.8 | 2 | 7 |

Table 5: Scaling experiments on FLUX.2-klein-base-4B with LoRA. TCA activates 30% of image tokens and all text tokens. Training and memory measurements use micro-batch 4 per GPU and 4-step accumulation on 4 A40 GPUs.

| Dataset | FID | | | CLIPScore | | | VRAM (GiB) | | Time (GPU-h) | | E-VRAM (GiB) | | BS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Base | LoRA | TCA | Base | LoRA | TCA | LoRA | TCA | LoRA | TCA | LoRA | TCA | LoRA | TCA |
| COCO | 44.73 | 36.28 | 36.85 | 34.85 | 34.77 | 34.77 | 104.0 | 63.7 | 5.89 | 4.85 | 74.7 | 34.5 | 7 | 16 |
| TextCaps | 40.69 | 32.69 | 33.13 | 36.35 | 35.78 | 35.75 | 104.0 | 63.7 | 5.90 | 4.86 | 74.7 | 34.5 | 7 | 16 |

tuning across three datasets. Increasing $K$ provides further but generally diminishing gains. Moreover, Fig. 3(b) shows that, although larger active subsets accelerate early optimization, the tested settings reach similar final training losses. Together, these observations indicate that propagating gradients through every image token is not essential for effective adaptation on vision transformer.

**Token selection strategy matters.** To examine the importance of gradient-active token selection, Fig. 4 compares TCA with ViT adaptations of TokenTune and TokenSeek. TokenTune selects tokens uniformly at random, whereas the TokenSeek uses precomputed attention- and gradient-based scores. Across the ViT variants, TCA achieves the highest fine-grained classification accuracy among the evaluated methods. Its average accuracy drop relative to full fine-tuning is only 0.4-0.7 percentage points, compared with 3.7-6.5 for TokenTune and 1.7-6.3 for TokenSeek. These results support TCA's use of the current completed forward pass to guide token selection.

**Full-sequence context matters.** Finally, we examine whether token merging or pruning methods Bolya et al. (2023); Lee et al. (2024); Mao et al. (2025); Marchetti et al. (2025) can achieve the same balance between training cost and accuracy as IEFT. As reported in Tab. 3, under comparable memory budgets, these approaches exhibit substantial accuracy degradation relative to TCA. These results support the importance of preserving full-sequence forward context during selective backpropagation. As also shown in Tab. 3, a practical combination is to apply TCA during training and token merging or pruning at inference. More ablation details are available in the Appendix.

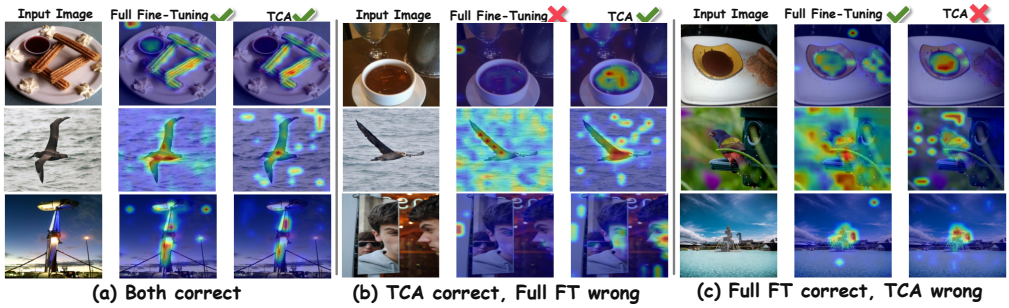



Figure 5: Attention maps from ViTs trained with full fine-tuning and TCA.

Table 6: Scaling experiments on Wan2.2-14B with LoRA. TCA activates 30% of video tokens and all text tokens. Training uses micro-batch 1 per GPU and 2-step accumulation on 8 B200 GPUs, while memory and BS measurements use that without checkpointing. For time measurements, full-token LoRA uses checkpointing, while TCA uses no checkpointing.

| Dataset | FVD-300 | | | Motion Smoothness | | Subject Consistency | | VRAM (GiB) | | Time (GPU-h) | | E-VRAM (GiB) | | BS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Base | LoRA | TCA | LoRA | TCA | LoRA | TCA | LoRA | TCA | LoRA | TCA | LoRA | TCA | LoRA | TCA |
| MSR-VTT | 817.2 | 580.4 | 590.8 | 96.0 | 96.0 | 86.5 | 87.1 | OOM | 692.5 | 9.85 | 4.85 | OOM | 476.8 | 0 | 2 |

### 4.5 Scaling to Multimodal Understanding and Visual Generation

Long multimodal and spatiotemporal sequences increase the token-dependent training costs targeted by TCA. We therefore evaluate TCA on Qwen3-VL-8B Bai et al. (2025), FLUX.2-klein-base-4B Labs (2025), and Wan2.2-14B Wan et al. (2025) under LoRA fine-tuning. Table 4 reports results across five multimodal benchmarks Lu et al. (2022); Marino et al. (2019); Liu et al. (2023a); Schwenk et al. (2022); Masry et al. (2022). Qwen3-VL maintains scores within 0.9 percentage points of full-token LoRA. Total VRAM/E-VRAM decrease by 21–31%/60–63%, while micro-batch capacity increases 3.5–5.0×. For image generation, Table 5 shows that FLUX.2 remains within 0.6 FID and 0.03 reported CLIPScore of full-token LoRA on COCO Lin et al. (2014) and TextCaps Sidorov et al. (2020). Total VRAM/E-VRAM decrease by 38.8%/53.8%, increasing micro-batch capacity from 7 to 16. For video generation, Table 6 reports FVD-300 of 590.8 for TCA versus 580.4 for full-token LoRA on MSR-VTT Xu et al. (2016), with comparable temporal quality. Under the same hardware, TCA achieves approximately 2× training throughput for Wan2.2-14B. Long sequences increase the absolute attention-related backward work that TCA can avoid. Memory savings also enable larger micro-batches. For video, avoiding checkpointing recomputation further contributes to the throughput gain. Details are provided in the Appendix.

### 4.6 Visualization and Limitations

Fig. 5 shows broadly similar ViT attention maps for TCA and full fine-tuning, with TCA's attention extending beyond the gradient-active subset. Although TCA exhibits more concentrated attention in some examples, the failures in Fig. 5(c) show that concentration does not necessarily improve recognition. A potential direction is to adaptively refine attention during inference for coverage.

### 4.7 Conclusion

Building on selective token back-propagation, we formulate input-efficient fine-tuning (IEFT) for vision transformers to reduce token-dependent training costs. We propose a two-phase caching-activation (TCA) strategy that uses information from a gradient-free full forward pass to select active tokens, then recomputes them against cached full-sequence context for back-propagation. Unlike token pruning or merging, TCA preserves full-sequence forward context while restricting gradient propagation to selected tokens. Experiments on 13 classification datasets demonstrate over 60% peak-memory reduction and approximately 45% training-time savings, with average accuracy degradation below 1%. TCA also complements PEFT, while extensions to Qwen3-VL, FLUX.2, and Wan2.2 demonstrate its applicability to multimodal understanding and visual generation.

## AI USE STATEMENT

The authors developed the research idea and initial method implementation, wrote the first manuscript draft, and conducted the main ViT experiments before using AI tools for subsequent assistance. ChatGPT and Codex were used to polish and revise the manuscript, improve the clarity of methodological and computational descriptions, and assist with locating and comparing related work. AI assistance also supported additional experiments, including implementation and debugging of model-specific extensions, preparation of experimental scripts and figures, and discussion of experimental configurations and results. Some revised passages were drafted with AI assistance and subsequently edited by the authors. The authors retained control over research decisions, experimental settings, and the selection of material included in the paper. The authors take full responsibility for the final manuscript, implementation, references, and reported experimental results.

## REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *International Conference on Learning Representations (ICLR)*, 2023.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision (ECCV)*, pp. 446–461, 2014.

Wenrui Cai, Qingjie Liu, and Yunhong Wang. Spmtrack: spatio-temporal parameter-efficient fine-tuning with mixture of experts for scalable visual tracking. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16871–16881, 2025.

Mengzhao Chen, Wenqi Shao, Peng Xu, Mingbao Lin, Kaipeng Zhang, Fei Chao, Rongrong Ji, Yu Qiao, and Ping Luo. Diffrate: Differentiable compression rate for efficient vision transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 17118–17128, 2023.

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

Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

Mir Rayat Imtiaz Hossain, Mennatullah Siam, Leonid Sigal, and James J Little. Visual prompting for generalized few-shot segmentation: A multi-scale approach. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23470–23480, 2024.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European Conference on Computer Vision (ECCV)*, pp. 709–727, 2022.

Shibo Jie and Zhi-Hong Deng. Fact: Factor-tuning for lightweight adaptation on vision transformer. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 37, pp. 1060–1068, 2023.

Shibo Jie, Zhi-Hong Deng, Shixuan Chen, and Zhijuan Jin. Convolutional bypasses are better vision transformer adapters. In *European Conference on Artificial Intelligence (ECAI)*, pp. 202–209, 2024a.

Shibo Jie, Yehui Tang, Jianyuan Guo, Zhi-Hong Deng, Kai Han, and Yunhe Wang. Token compensator: Altering inference cost of vision transformer without re-tuning. In *European Conference on Computer Vision (ECCV)*, pp. 76–94, 2024b.

Samar Khanna, Medhanie Irgau, David B Lobell, and Stefano Ermon. Explora: Parameter-efficient extended pre-training to adapt vision transformers under domain shifts. In *International Conference on Machine Learning (ICML)*, 2025.

Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. CVPR workshop on fine-grained visual categorization (FGVC)*, volume 2, 2011.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D object representations for fine-grained categorization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 554–561, 2013.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Black Forest Labs. FLUX.2: Frontier Visual Intelligence. `https://bfl.ai/blog/flux-2`, 2025.

Sanghyeok Lee, Joonmyung Choi, and Hyunwoo J. Kim. Multi-criteria token fusion with one-step-ahead attention for efficient vision transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Sheng Li, Qitao Tan, Yue Dai, Zhenglun Kong, Tianyu Wang, Jun Liu, Ao Li, Ninghao Liu, Yufei Ding, Xulong Tang, and Geng Yuan. Mutual effort for efficiency: A similarity-based token pruning for vision transformers in self-supervised learning. In *International Conference on Learning Representations (ICLR)*, pp. 32226–32243, 2025.

Tao Li, Yingqi Gao, Kai Wang, Song Guo, Hanruo Liu, and Hong Kang. Diagnostic assessment of deep learning algorithms for diabetic retinopathy screening. *Information Sciences*, 501:511–522, 2019.

Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Scaling & shifting your features: A new baseline for efficient model tuning. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:109–123, 2022.

Youwei Liang, Chongjian GE, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. *International Conference on Learning Representations (ICLR)*, 2022.

Peiyuan Liao, Xiuyu Li, Xihui Liu, and Kurt Keutzer. The artbench dataset: Benchmarking generative models with artworks, 2022.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pp. 740–755, 2014.

Fangyu Liu, Guy Emerson, and Nigel Collier. Visual spatial reasoning. *Transactions of the Association for Computational Linguistics*, 11:635–651, 2023a.

Weihuang Liu, Xi Shen, Chi-Man Pun, and Xiaodong Cun. Explicit visual prompting for low-level structure segmentations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19434–19445, 2023b.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9992–10002, 2021.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *International Conference on Learning Representations (ICLR)*, 2019.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems (NeurIPS)*, 35: 2507–2521, 2022.

Lei-Lei Ma, Shuo Xu, Ming-Kun Xie, Lei Wang, Dengdi Sun, and Haifeng Zhao. Correlative and discriminative label grouping for multi-label visual prompt tuning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 25434–25443, 2025.

Junzhu Mao, Yang Shen, Jinyang Guo, Yazhou Yao, Xiansheng Hua, and Hengtao Shen. Prune and merge: Efficient token compression for vision transformer with spatial information preserved. *IEEE Transactions on Multimedia (TMM)*, 27:4670–4683, 2025.

Michele Marchetti, Davide Traini, Domenico Ursino, and Luca Virgili. Efficient token pruning in vision transformers using an attention-based multilayer network. *Expert Systems with Applications*, 279:127449, 2025.

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3190–3199, 2019.

Ahmed Masry, Do Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL*, pp. 2263–2279, 2022.

Sachin Mehta and Mohammad Rastegari. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. In *International Conference on Learning Representations (ICLR)*, 2022.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pp. 722–729, 2008.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182, 2023.

Rong Qin, Xin Liu, Xingyu Liu, Jiaxuan Liu, Jinglei Shi, Liang Lin, and Jufeng Yang. No pains, more gains: Recycling sub-salient patches for efficient high-resolution image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14965–14975. IEEE, 2025.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:13937–13949, 2021.

Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. A-OKVQA: A benchmark for visual question answering using world knowledge. In *European Conference on Computer Vision (ECCV)*, pp. 146–162, 2022.

Oleksii Sidorov, Ronghang Hu, Marcus Rohrbach, and Amanpreet Singh. Textcaps: a dataset for image captioning with reading comprehension. In *European Conference on Computer Vision (ECCV)*, pp. 742–758, 2020.

Antoine Simoulin, Namyong Park, Xiaoyi Liu, and Grey Yang. Memory-efficient fine-tuning of transformers via token selection. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 21565–21580, 2024.

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5217–5227, 2022.

Wenhao Tang, Sheng Huang, Qiming Zhao, Ren Li, and Luwen Huangfu. An iteratively optimized patch label inference network for automatic pavement distress detection. *IEEE transactions on intelligent transportation systems*, 23(7):8652–8661, 2021.

Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 595–604, 2015.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Hongjie Wang, Bhishma Dedhia, and Niraj K Jha. Zero-tprune: Zero-shot token pruning through leveraging of the attention graph in pre-trained transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16070–16079, 2024.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025.

Jun Xu, Tao Mei, Ting Yao, and Yong Rui. Msr-vtt: A large video description dataset for bridging video and language. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5288–5296, 2016.

Quanzeng You, Jiebo Luo, Hailin Jin, and Jianchao Yang. Building a large scale dataset for image emotion recognition: The fine print and the benchmark. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 30, 2016.

Runjia Zeng, Qifan Wang, Qiang Guan, Ruixiang Tang, Lifu Huang, Zhenting Wang, Xueling Zhang, Cheng Han, and Dongfang Liu. Tokenseek: Memory efficient fine tuning via instance-aware token ditching. *arXiv preprint arXiv:2601.19739*, 2026.

Wang Zeng, Sheng Jin, Wentao Liu, Chen Qian, Ping Luo, Wanli Ouyang, and Xiaogang Wang. Not all tokens are equal: Human-centric visual analysis via token clustering transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11091–11101, 2022.

Yuanhan Zhang, Kaiyang Zhou, and Ziwei Liu. Neural prompt search. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 47(7):5268–5280, 2024.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025.

# A Appendix

## A.1 Additional Implementation Details for ViT Experiments

**Token selection and replay.** During the gradient-free caching phase, TCA processes the complete token sequence and stores layer-wise keys and values. Image tokens are ranked by the final-layer attention from the classification token, averaged across heads. The top 32 image tokens are selected

Table 7: Dataset sizes used in the scaling experiments. Counts refer to the examples used in our experiments. ScienceQA includes only image-containing examples; VSR uses the random split.

| Dataset | Training | Evaluation |
|---|---|---|
| *Multimodal understanding* | | |
| ScienceQA Lu et al. (2022) | 6,218 | 2,097 |
| OK-VQA Marino et al. (2019) | 9,009 | 5,046 |
| VSR Liu et al. (2023a) | 7,680 | 1,097 |
| A-OKVQA Schwenk et al. (2022) | 17,056 | 1,145 |
| ChartQA Masry et al. (2022) | 28,299 | 1,920 |
| *Image generation* | | |
| COCO Lin et al. (2014) | 10,000 | 2,000 |
| TextCaps Sidorov et al. (2020) | 10,000 | 2,000 |
| *Video generation* | | |
| MSR-VTT Xu et al. (2016) | 6,513 | 300 |

independently for each input, and the classification token is always retained. The selected positions are shared across all replay layers and recomputed for each training input. Both phases use the same augmented image, with no parameter update between caching and replay. During replay, active-token keys and values are recomputed with gradients and replace the corresponding cached entries at their original positions. Unselected entries remain detached. Consumed cache references are released layer by layer; tensors required by autograd remain available for back-propagation. The classification loss is computed from the replayed classification-token representation.

**Adaptation of token-selection baselines.** TokenTune and TokenSeek are adapted to ViT with $K = 33$, including the classification token. TokenTune randomly samples 32 image tokens uniformly without replacement. TokenSeek instead selects 32 image tokens using precomputed attention- and gradient-based importance scores that remain fixed during fine-tuning. Both retain the classification token and use the same selected positions across layers.

**Comparison and Combination with Token Reduction.** We evaluate all methods on CIFAR-100 and Food-101 using ViT-S/16 and ViT-B/16 at $224 \times 224$ resolution. All models are initialized from ImageNet-pretrained checkpoints and fine-tuned for 70 epochs. Unless otherwise stated, ToMe, TRAM, and TCA use AdamW with a learning rate of $3 \times 10^{-5}$, a batch size of 128, and random seed 42. TCA uses dataset-specific normalization for CIFAR-100 and Food-101. ToMe uses the official timm patch with $r = 32$, CLS-token protection, proportional attention, and size-weighted token merging between the attention and MLP sublayers. MCTF uses the official implementation with AdamW, a learning rate of $1.5 \times 10^{-5}$, weight decay $0.05$, cosine decay, seed 0, and EMA decay $0.99996$. We set `mctf_type=[64,8,0,1,1,1,20,40,1,1,0]`. For PM-ViT, we use the attention-guided configuration with pruning rate $0.1$ and keep rate $0.5$ across all 12 blocks. The learning rate is $10^{-4}$, with a minimum of $10^{-6}$, weight decay $0.05$, and five warm-up epochs. Distillation, Mixup, CutMix, random erasing, and EMA are disabled. The standalone TRAM baseline retains 44 patch tokens per block and computes token importance using cumulative maximum-head attention centrality. For TCA+TRAM, training uses TCA with `top_k=32`, corresponding to 32 active patch tokens plus CLS. TRAM is applied only at inference. The CLS token is always preserved, and no additional fine-tuning is performed.

### A.2 Details of Multimodal Understanding and Visual Generation

**Models and adaptation.** We evaluate Qwen3-VL-8B, the non-distilled FLUX.2-klein-base-4B, and Wan2.2-T2V-A14B using LoRA, with pretrained weights frozen. Throughout these experiments, full-token LoRA denotes back-propagation through all tokens with the same trainable adapters, rather than full-parameter fine-tuning.

**Multimodal understanding.** For Qwen3-VL, LoRA is applied to the language decoder's query, key, value, output, and MLP projections, with rank 16, scaling parameter 32, and zero dropout. The

visual encoder remains frozen. TCA selects 30% of candidate image tokens and 30% of candidate prompt-text tokens. Image-token importance is obtained from attention issued by post-image question tokens, averaging across heads and taking the maximum across query positions. Prompt-text importance is obtained from answer-prediction queries. All supervised answer tokens and their causal prediction positions remain active; consequently, the overall active fraction can exceed 30%. During replay, recomputed active-token keys and values replace the corresponding cached entries at their original positions. We use the splits summarized in Table 7. Evaluation uses greedy generation and dataset-specific scoring: multiple-choice accuracy for ScienceQA and A-OKVQA, binary accuracy for VSR, soft VQA scoring for OK-VQA, and relaxed accuracy for ChartQA, with 5% numerical tolerance and normalized exact matching otherwise.

**Image generation.** For each of COCO and TextCaps, we use fixed subsets of 10,000 training images and 2,000 held-out evaluation images, with one fixed caption per image. Images are resized using bicubic interpolation and center-cropped to $512 \times 512$, without random flipping. Frozen VAE latents and text-encoder features are precomputed. Text features use Qwen3 hidden layers 9, 18, and 27 with sequence length 128; image latents use the VAE posterior mode and pretrained normalization. LoRA uses rank/scaling 16/16 and zero dropout on attention, MLP, and image-output projections. We train for 1,000 optimizer updates using AdamW with learning rate $5 \times 10^{-5}$, weight decay 0.01, $(\beta_1, \beta_2) = (0.9, 0.999)$, and gradient clipping at 1. The learning rate increases during 100 warmup updates and remains constant thereafter. We use seed 42 and caption dropout 0.1. Training uses four A40 GPUs, micro-batch size 4 per GPU, and four-step gradient accumulation, giving an effective batch size of 64. Pretrained weights use BF16, while adapters and optimizer states use FP32. Gradient checkpointing is disabled, and both methods use the same FlashAttention backend through PyTorch SDPA.

TCA employs spatially stratified, loss-guided token selection. During the gradient-free full forward pass, we compute each image token's velocity-prediction MSE, averaged over channels, using the current noise and timestep. The $32 \times 32$ image-token grid is partitioned into $14 \times 22$ non-overlapping spatial regions, and the highest-loss token in each region is selected. This yields 308 gradient-active image tokens while ensuring coverage across regions. All 128 text tokens remain active. The selected indices are shared across replay layers and recomputed for each training example at each iteration. Scoring and selection are performed without gradient computation, reusing the cached backbone outputs with an additional output-head evaluation. The 30% budget therefore applies only to image tokens. The final flow-matching loss is evaluated only at selected image positions, using noise minus normalized latent as the target. Token selection therefore restricts both backward computation and the positions receiving direct supervision, while preserving full-sequence forward context. Evaluation uses dense inference at $512 \times 512$, 50 sampling steps, guidance scale 4.0, and paired fixed seeds. We compute FID-2K using clean-fid's clean Inception-v3 features and CLIPScore as 100 times paired cosine similarity using `laion/CLIP-ViT-B-32-laion2B-s34B-b79K`.

**Video generation.** We fine-tune Wan2.2-T2V-A14B on 6,513 MSR-VTT training videos and evaluate on a fixed set of 300 held-out videos. We use one caption per video. Each clip contains 33 frames sampled at 8 fps. Frames are resized and center-cropped to $480 \times 832$. Frozen VAE latents and UMT5 text features are precomputed. Text sequences are capped at 512 tokens. We train the high- and low-noise LoRA experts separately for 500 optimizer updates each. LoRA uses rank/scaling 16/16 and zero dropout. We use AdamW with learning rate $5 \times 10^{-5}$, weight decay 0.01, and gradient clipping at 1. Training uses eight B200 GPUs, micro-batch 1 per GPU, and two-step gradient accumulation, yielding an effective batch size of 16. The recorded quality-training configuration enables activation checkpointing. Pretrained transformer weights use BF16; adapters and optimizer states use FP32.

TCA retains full video and text context in a gradient-free forward pass, then replays selected video-token paths. It activates 30% of video tokens; text conditioning is not sampled. The flow-matching loss is averaged over selected positions, using noise minus normalized latent as the target. Efficiency is measured separately from quality training. Memory is measured at micro-batch 1 without checkpointing. For the throughput comparison, full-token LoRA uses checkpointing at micro-batch 4, whereas TCA uses no checkpointing at micro-batch 2; neither uses accumulation. Evaluation uses the same 300 prompts and paired seeds for all methods, with dense 33-frame inference at $480 \times 832$ and 40 sampling steps. FVD-300 compares the feature distributions of the 300 generated clips and

their real counterparts. We compute Motion Smoothness and Subject Consistency using VBench on these generated clips. Motion Smoothness compares AMT-S-interpolated intermediate frames with observed frames. Subject Consistency averages DINO feature similarity to the first and preceding frames. Both VBench scores are averaged over clips and multiplied by 100 for reporting.