big data and cognitive computing

MDPI

Article

Joint MLP and Token Pruning for Personalizing Vision Transformers

Zhiyue Li, Tong Liu *, Feng Huang, Xinzhi Huang and Zhihao Zou

School of Computer and Network Security, Chengdu University of Technology, Chengdu 610059, China; lizhiyue@stu.cdut.edu.cn (Z.L.); huangfeng691@stu.cdut.edu.cn (F.H.); huangxinzhi@stu.cdut.edu.cn (X.H.); zouzhihao@stu.cdut.edu.cn (Z.Z.)

* Correspondence: liutong1982@cdut.edu.cn

Abstract

ViTs have achieved excellent performance in image recognition tasks, but their large parameter counts and high computational complexity limit their deployment on resource-constrained devices. Most existing ViT pruning methods adopt class-agnostic pruning strategies, which fail to distinguish the diverse structural requirements of different target classes. As a result, they are prone to removing critical features, leading to class-wise accuracy imbalance in practical deployment. To address this issue, this paper proposes a class-aware joint pruning framework for ViTs, which collaboratively compresses the model from two orthogonal dimensions: MLP neurons and visual tokens. Specifically, (1) based on first-order Taylor expansion, we quantify the contribution of each MLP neuron to the target classes and adaptively prune redundant neurons to achieve structured compression, followed by lightweight fine-tuning on the target class subset; (2) we propose a Class-Guided Token Selection (CGTS) method, which constructs class prototype vectors using a few support samples of the target classes and then dynamically selects patch tokens that are semantically highly relevant to the target classes during inference in a zero-shot manner, requiring no additional training or fine-tuning. The two modules complement each other, achieving dual compression from the parameter dimension and the inference data dimension. Experiments on CIFAR-100 and TinyImageNet datasets using DeiT-Tiny/Small models demonstrate that, compared with state-of-the-art pruning methods, our method reduces GMACs on target class subsets by up to 48%, improves inference speed by nearly 50%, and requires only 0.8 KB of additional storage overhead per subset, ultimately achieving a superior trade-off among accuracy, computational efficiency, and storage overhead.

Keywords: vision transformer; class-aware pruning; efficient inference; model compression

Check for updates

Academic Editor: Francesco Cauteruccio

Received: 26 May 2026
Revised: 2 August 2026
Accepted: 6 August 2026
Published: 9 August 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

The rapid advancement of artificial intelligence (AI) has driven profound transformations in computer vision. Vision Transformer (ViT) [1], benefiting from its global self-attention mechanism and strong representation learning capability, has achieved superior performance over traditional convolutional neural networks (CNNs) in image classification [2–4], object detection [5–8], and semantic segmentation [9–11]. However, ViT models have enormous parameters and thus suffer from high computational complexity, which hinders their efficient lightweight deployment on resource-constrained devices [12].

Model pruning [13,14], as a widely used deep learning lightweight technique, removes redundant parameters and structures in neural networks to compress complex models and accelerate inference with minimal accuracy degradation. Most early research on model pruning focused on CNNs, which remove network redundancy by pruning local structures such as convolutional kernels and feature channels. With the widespread adoption of ViTs in computer vision, ViT pruning has gradually emerged as a new research hotspot. Unlike the pruning paradigm of CNNs, which only exploits local feature redundancy, Vision Transformers possess inherent global modeling capability, giving rise to dedicated pruning strategies and optimization schemes distinct from those designed for CNNs. Existing ViT pruning methods, including structured pruning [15–17] and token pruning [18–20], can effectively reduce model complexity and computational overhead.

To ensure their generality, models are typically trained with enormous data, allowing them to classify diverse classes. However, in practical deployment scenarios, models are usually used in a specific environment and only perceive a limited number of classes. Then, many components in a fully trained model that are not relevant to these predicted classes are redundant. Hence, in recent years, researchers have proposed class-aware pruning methods for CNNs. By accurately identifying the relevant components of different classes, these methods selectively retain key features of target classes and eliminate redundant structures irrelevant to these classes. They can compress models to improve performance while maintaining high accuracy. Representative methods such as CAPTOR [21], CRISP [22], and OCAP [23] have achieved promising results for CNNs.

However, these methods proposed for CNNs [24,25] are not directly applicable to ViTs, as ViTs introduce distinct pruning dimensions, such as tokens and multi-head attentions. In addition, the relevance of these new pruning dimensions to class-aware pruning remains largely unexplored. Most existing ViT pruning methods adopt a class-agnostic paradigm and follow a globally unified pruning criterion. They fail to distinguish the important features of target classes, likely removing the features related to target classes and hurting the final accuracy. As demonstrated in CNNs, class-aware pruning can reduce model complexity and memory footprint while retaining or improving prediction accuracy. Thus, it is worth exploring class-aware pruning in ViTs, as ViTs are gradually replacing CNNs to be the backbones of many CV tasks.

To the best of our knowledge, this paper is among the first to simultaneously investigate class-aware joint compression of ViTs from the two orthogonal dimensions (MLP neurons and visual tokens) and propose a corresponding joint pruning framework. Structured MLP pruning has been extensively studied for reducing model parameters, yet existing approaches lack adaptation to target classes. Token pruning has been considered a promising technique to reduce model complexity for transformer-based models, such as ViTs and large language models (LLMs). However, we find that many token pruning methods that use gates to control token propagation actually cannot improve a model's performance due to the extra overhead caused by the control gate. In this paper, we propose a lightweight and zero-shot pruning method tailored for class-aware scenario.

The main contributions of this paper are as follows:

- We observe that MLP modules in ViTs are highly class-sensitive. Therefore, we propose a class-aware structured MLP pruning method based on first-order Taylor expansion, which computes the importance scores of neurons with respect to target classes and adaptively prunes redundant neurons, thereby achieving effective compression of the MLP modules, which requires lightweight fine-tuning on the target class subset to recover minor accuracy loss. Since the neurons in MLPs are pruned structuredly, dense patterns in MLPs are reserved and no special accelerator is required. This pruning

stage compresses the feature space and provides a low-noise backbone network for subsequent token-level optimization.

- Building on MLP pruning, we propose zero-shot training-free Class-Guided Token Selection (CGTS), which filters class-relevant tokens via prototype vectors and adaptively determines the pruning layer and retention ratio without manual tuning. The two modules form a unified joint framework via progressive fusion, delivering orthogonal complementary gains across the parameter and inference dimensions.
- We conduct comprehensive experiments to evaluate our proposed method compared to some state-of-the-art methods. Experimental results demonstrate that, while preserving accuracy, our method reduces Giga Multiply-Accumulate Operations (GMACs) by up to 48%, improves inference speed by nearly 50%, and incurs only 0.8 KB of additional storage overhead per subset. Compared with existing pruning methods, our method achieves a better balance between computational cost, storage overhead, and inference latency.

The remainder of this paper is organized as follows. Section 2 covers related work. Section 3 details the CGTS and MLP pruning methods and their synergistic integration. Section 4 presents the experimental results, and Section 5 concludes this paper. Our code is available at https://github.com/lee95783/class_aware (accessed on 2 August 2026).

2. Related Work

Pruning ViT models is primarily achieved through token pruning and structured pruning. Token pruning reduces the quadratic complexity of self-attention by dynamically or statically discarding uninformative tokens. DynamicViT [18] inserts lightweight prediction modules between Transformer blocks to learn token retention probabilities end-to-end and progressively prunes redundant tokens, while training-free methods such as Zero-TPrune [26] leverage a weighted PageRank algorithm on attention graphs to assess token importance in a zero-shot setting. On the other hand, MLP modules bear the primary burden of feature transformation in ViTs and are thus the central target of structured pruning; approaches like AMP [27] and DGMR [28] achieve substantial parameter reduction by evaluating neuron importance. Moreover, SViTE [29] attempts to cover both token and structural dimensions within a unified framework by jointly imposing sparsity constraints on weights, attention heads, and tokens during training.

While these methods have distinct optimization directions, they all follow a task-agnostic pruning paradigm: the token retention decisions of DynamicViT [18] and Zero-TPrune [26] are independent of specific target classes, and the joint sparsity constraints of SViTE [29] are not customized for particular class subsets, making it impossible to achieve fine-grained targeted sparsification. Furthermore, except for a few training-free approaches, the vast majority still require expensive full-scale fine-tuning for each pruning configuration.

In contrast to task-agnostic strategies, class-aware pruning seeks to incorporate target-class priors into pruning decisions, enabling compressed models to focus more sharply on task-relevant discriminative features. Early efforts predominantly targeted CNN architectures: Hemmat et al. [30] employed a statistically driven "class-specific excitation rate" to prune neurons ineffective for the target classes; Qin et al. [21] clustered filters with similar response patterns to remove filter groups mismatched with the target classes; and Hu et al. [31] further formulated channel pruning as a submodular set function maximization problem, optimizing inter-class and intra-class scatter via the trace ratio criterion. Subsequent work extended class-aware pruning to practical deployment efficiency [22], 3D point cloud tasks [32], and on-device dynamic scenarios [23]. Although these studies have advanced the development of class-aware pruning, they all rest on the homogeneous

structural assumptions of CNNs, and their filter-oriented importance metrics cannot be readily transferred to the heterogeneous components of ViTs, such as MLPs and tokens.

Recently, class-aware ideas have been preliminarily introduced into ViT lightweighting. X-Pruner [33] learns class-level explainability masks for each prunable unit, retaining structured components critical to the target classes through end-to-end training. Vulcan [34] reveals from a knowledge-disentanglement perspective that FFNs encode class-specific knowledge and proposes a "train-then-prune" paradigm to generate class-customized ViT models. However, these methods focus exclusively on structured component pruning without addressing token-level redundancy, and they still rely on full training or fine-tuning procedures, incurring substantial training overhead.

However, these methods still exhibit clear limitations: X-Pruner [33] adopts a soft gating mechanism that fails to achieve genuine structural compression and inference speedup; Vulcan [34] only performs structured pruning on feed-forward network (FFN) modules and completely neglects token-level redundancy. More critically, both methods rely on full end-to-end training, leading to high deployment costs for new class subsets.

To sum up, three key limitations remain unresolved in existing ViT pruning methods: (1) most methods follow a task-agnostic paradigm and cannot achieve targeted compression for specific class subsets; (2) existing class-aware ViT pruning only covers a single structural dimension, without joint optimization of MLP parameters and visual tokens; and (3) the vast majority of schemes depend on full-scale fine-tuning and lack lightweight zero-shot deployment solutions.

We propose a class-aware dual-dimension synergistic compression framework for Vision Transformers. Our work simultaneously covers both token and structural dimensions within a unified class-aware framework, addressing the methodological gap in existing studies that follow a task-agnostic paradigm or only optimize a single dimension. The entire framework avoids full-scale retraining, requiring only minimal fine-tuning on the target class subset for deployment adaptation, with training overhead substantially lower than that of existing full-scale fine-tuning schemes. The two compression dimensions operate in a progressively synergistic manner, effectively reducing edge-side inference latency while preserving accuracy on the target classes.

3. Proposed Method

This section presents the details of our proposed method. The overview of the method is shown in Figure 1. Overall, we adopt a backbone-first then fine-grained optimization strategy. After completing MLP backbone pruning and determining the fundamental network structure, we further conduct fine-grained pruning at the patch token level. The pruning pipeline starts with two offline tasks, as described in Section 3.1: class-aware pruning of MLP modules based on first-order Taylor expansion, followed by lightweight fine-tuning of the pruned model, thereby achieving the compression and reconstruction of the model backbone. Class prototypes are then constructed offline, and pruning layers are adaptively selected on the compressed model, followed by dynamic token screening in the online inference stage, as elaborated in Section 3.2. The progressive fusion and collaborative compression strategy of the two pruning schemes are explained in Section 3.3.

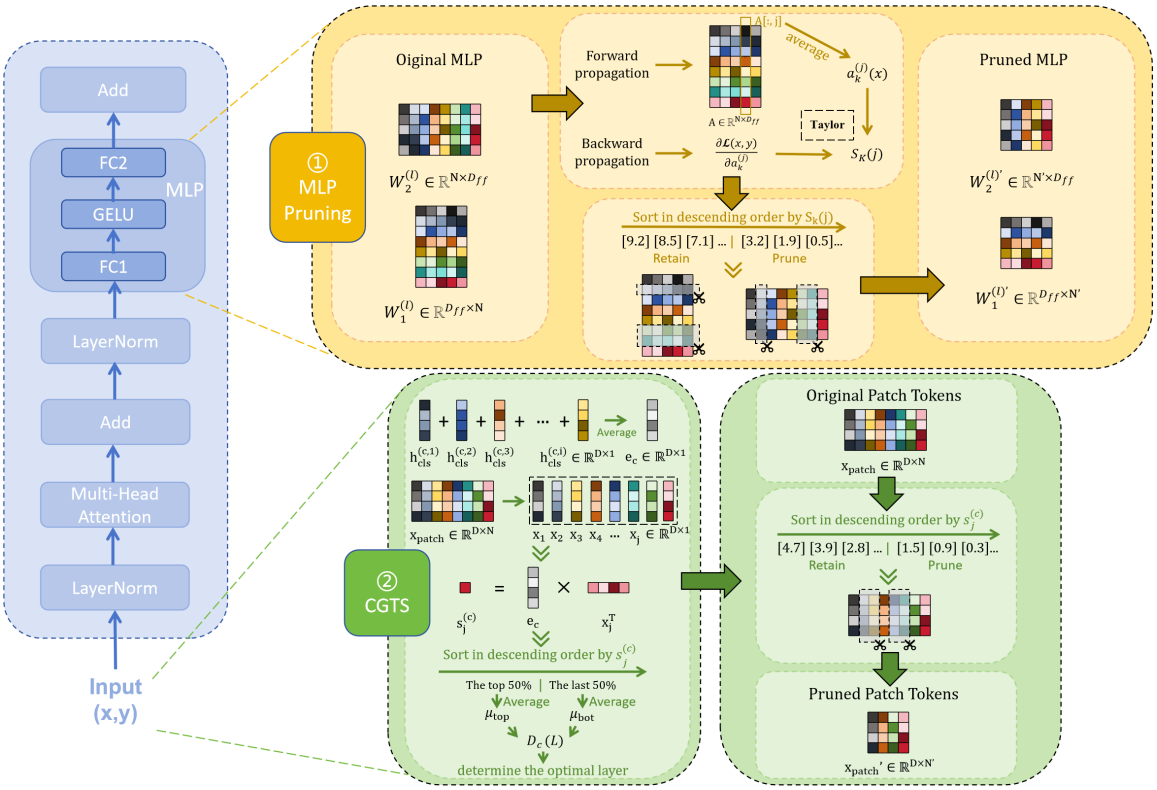


Figure 1. Overview of the MLP pruning and CGTS framework.

3.1. MLP Pruning

In Vision Transformer architectures, multi-head attention and MLP feed-forward networks exhibit distinct differences in pruning adaptability. Attention heads occupy a low proportion of total parameters and present class-agnostic properties, resulting in limited pruning gains. Experimental results further demonstrate that pruning attention heads can barely reduce inference overhead and is prone to inducing significant accuracy degradation when numerous classes are retained. In contrast, MLP modules have high parameter and computational ratios with prominent neuron redundancy. Meanwhile, they possess strong class-aware modeling ability and deliver remarkable pruning potential and benefits, which makes them our primary pruning target.

To quantify the contribution of each MLP neuron to the target class, we adopt the first-order Taylor expansion to approximate the impact of neuron removal on the loss function. Compared with weight-magnitude-based pruning criteria, the first-order Taylor approximation more accurately reflects each neuron's contribution to model outputs. Meanwhile, unlike Hessian-based pruning that depends on second-order curvature, it requires only a small set of samples for low-overhead importance ranking, with better numerical stability and lower deployment costs on target class subsets. For the j-th neuron in the k-th MLP layer, its importance score is defined as the average absolute value of the product of activation and gradient computed on the target class subset $\mathcal{T}_c$:

$$S_k(j) = \frac{1}{|\mathcal{T}_c|} \sum_{(\mathbf{x},y)\in\mathcal{T}_c} \left| f_k^j(\mathbf{x}) \cdot \frac{\partial \mathcal{L}(\mathbf{x},y)}{\partial f_k^j(\mathbf{x})} \right| \tag{1}$$

where $f_k^j(\boldsymbol{x})$ denotes the activation value of input $\boldsymbol{x}$ at the corresponding neuron, and $\mathcal{L}$ represents the cross-entropy loss function. This score jointly captures the activation intensity of neurons and their sensitivity to the loss function. A higher score indicates a greater contribution of the neuron to target class recognition. The entire calculation only

requires limited forward and backward propagation on the target class subset, without accessing the full training dataset.

After obtaining the importance scores of all neurons in each layer, we determine the number of reserved neurons according to a predefined pruning ratio $\alpha \in (0,1)$. Specifically, for the k-th layer, we sort the N_k neurons in descending order based on $S_k(j)$ and retain the top $\lceil (1-\alpha) \cdot N_k \rceil$ high-contribution neurons while discarding the remaining redundant ones. Accordingly, we reconstruct compact weight matrices with reduced dimensions. The original input weight matrix $\boldsymbol{W}_1^k \in \mathbb{R}^{N_k \times D}$ is replaced by $\boldsymbol{W}_1^{k\prime} \in \mathbb{R}^{\lceil (1-\alpha) N_k \rceil \times D}$, and the output weight matrix $\boldsymbol{W}_2^k \in \mathbb{R}^{D \times N_k}$ is updated to $\boldsymbol{W}_2^{k\prime} \in \mathbb{R}^{D \times \lceil (1-\alpha) N_k \rceil}$. We migrate the weights corresponding to reserved neurons from the original matrices to the newly constructed ones. This operation effectively reduces model parameters and FLOPs, leading to lower inference latency under standard inference frameworks.

Finally, we perform minimal fine-tuning on the pruned compact model using the target class subset to compensate for the slight accuracy degradation caused by structured pruning. We summarize the class-aware MLP pruning pipeline in Algorithm 1.

Algorithm 1 Class-aware MLP structured pruning.

Require: Pre-trained ViT $\mathcal{M}$ with MLP layers; Weights $\mathbf{W}_1^k, \mathbf{W}_2^k$ and neuron number N_k for layer k; Target class dataset $\mathcal{T}_c$; Pruning ratio $\alpha \in (0,1)$; Fine-tune epochs E
Ensure: Compact pruned & fine-tuned model $\mathcal{M}_{\text{pruned}}$

for each MLP layer k of $\mathcal{M}$ **do**
  for each neuron j in layer k **do**
    $S_k(j) \leftarrow \frac{1}{|\mathcal{T}_c|} \sum_{(\mathbf{x},y)\in\mathcal{T}_c} \left| f_k^j(\mathbf{x}) \cdot \frac{\partial \mathcal{L}(\mathbf{x},y)}{\partial f_k^j(\mathbf{x})} \right|$
  end for
  Sort $\{S_k(j)\}_{j=1}^{N_k}$ in descending order
  $M \leftarrow \lceil (1-\alpha) \cdot N_k \rceil$ ▷ Number of reserved neurons
  idx $\leftarrow$ Top-M indices of sorted scores
  $\mathbf{W}_1^{k\prime} \leftarrow \mathbf{W}_1^k[\text{idx}, :]$ ▷ Slice input weight rows
  $\mathbf{W}_2^{k\prime} \leftarrow \mathbf{W}_2^k[:, \text{idx}]$ ▷ Slice output weight columns
  Update $\mathbf{W}_1^k \leftarrow \mathbf{W}_1^{k\prime}$, $\mathbf{W}_2^k \leftarrow \mathbf{W}_2^{k\prime}$
end for
for epoch $= 1$ to E **do**
  Fine-tune $\mathcal{M}_{\text{pruned}}$ on $\mathcal{T}_c$ with cross-entropy loss
end for
return $\mathcal{M}_{\text{pruned}}$

3.2. CGTS

CGTS consists of two components: layer selection and token selection. Token pruning is performed only once: once some patch tokens are removed at a certain layer, all subsequent layers benefit. The selection of the optimal pruning layer requires a trade-off between semantic richness and speedup gain, and varies with the subset size K. Therefore, we first select the optimal layer offline in Section 3.2.1 and then dynamically select tokens at this layer using the class prototype in Section 3.2.2. The complete workflow of CGTS is summarized in Algorithm 2.

3.2.1. Layer Selection

It is crucial to balance accuracy and inference efficiency when selecting the pruning layer L for both single-class and multi-class tasks. If L is set too shallow, features are not fully aggregated, class prototypes lack semantic discrimination, and similarity scores are nearly random, easily causing erroneous removal of critical tokens. If L is set too deep, most computational overhead has already been incurred, resulting in limited compression gains.

Algorithm 2 CGTS.

Require: Target class set $\mathcal{C}$, candidate layers $\mathcal{L}_{\text{cand}}$, minimum layer $L_{\min}$, support sample number N_c, retention ratio ρ, input token sequence $\mathbf{t}$
Ensure: Pruned token sequence $\mathbf{t}_{\text{pruned}}$
Offline: Adaptive Layer Selection
for $L \in \mathcal{L}_{\text{cand}}$ **do**
  Compute class-wise prototype:
  $\mathbf{e}_c^{(L)} = \frac{1}{N_c} \sum_{i=1}^{N_c} \mathbf{h}_{\text{CLS}}^{(c,i)}$
  Calculate layer discriminability $D(L)$ via Equation (2)
end for
$\tau \leftarrow \left(\min_{L \in \mathcal{L}_{\text{cand}}} D(L) + \max_{L \in \mathcal{L}_{\text{cand}}} D(L)\right)/2$
$L^* \leftarrow \min\left\{L \mid L \geq L_{\min},\ D(L) > \tau\right\}$
$\mathbf{e}_{\text{shared}} \leftarrow \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \mathbf{e}_c^{(L^*)}$
Inference: Class-Guided Token Selection
Extract patch tokens $\{\mathbf{x}_j\}$ at input of layer L^*, $N \leftarrow |\{\mathbf{x}_j\}|$
for $j = 1$ to N **do**
  $s_j(c) \leftarrow \mathbf{x}_j^\top \mathbf{e}_{\text{shared}}$
end for
Sort $\{s_j(c)\}$ in descending order
$\mathcal{I} \leftarrow$ indices of top $\lceil \rho N \rceil$ scores
Sort $\mathcal{I}$ in ascending order
$\mathbf{t}_{\text{pruned}} \leftarrow \left[\mathbf{h}_{\text{CLS}}; \mathbf{x}_j \mid j \in \mathcal{I}\right]$
return $\mathbf{t}_{\text{pruned}}$

To achieve adaptive optimal pruning layer selection, we propose a prototype discriminability-based layer selection strategy, which only relies on support samples and does not require test set labels. The core goal is to preferentially select layers with strong capability to distinguish effective semantic tokens from irrelevant background tokens, so as to accurately remove redundant tokens and preserve critical semantic information. To quantitatively measure the semantic discrimination significance of different layers, we use the score difference between high-quality and low-quality tokens to represent the basic discrimination amplitude and adopt standard deviation normalization to eliminate feature scale differences across layers, enabling fair comparison of discriminability and constructing a standardized prototype discriminability score. Specifically, we extract patch token features of support samples at the input of candidate layer L and calculate their similarity scores with $\mathbf{e}_c$. For each image, all scores are sorted in descending order, where μ_{top} denotes the mean of the top 50% scores, μ_{bot} denotes the mean of the bottom 50% scores, and σ denotes the standard deviation of all scores. The prototype discriminability score of layer L for class c is defined as

$$D_c(L) = \frac{\mu_{\text{top}} - \mu_{\text{bot}}}{\sigma} \tag{2}$$

A larger $D_c(L)$ indicates that the layer can better distinguish class-relevant tokens from irrelevant ones. For multi-class tasks, the comprehensive layer discriminability score is calculated as $D(L) = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} D_c(L)$, representing the average discrimination capability for all target classes. After obtaining $D(L)$ for all candidate layers, the optimal pruning layer L^* is determined as follows:

$$L^* = \min\{L \mid L \geq L_{\min} \text{ and } D(L) > \tau\} \tag{3}$$

where $L_{\min}$ is the minimum allowable pruning layer, and τ is set to the midpoint of the value range of all candidate $D(L)$. This rule skips unreliable shallow layers and selects the earliest layer with discriminability above the average level, maximizing computational savings under controllable accuracy loss.

3.2.2. Token Selection

ViT divides an input image into a fixed number of patch tokens, together with a class (CLS) token for classification. In practical inference, only a small number of patch tokens carry discriminative semantic information to support the final prediction, while most background or low-information tokens contribute little to the classification decision. This motivates us to dynamically prune redundant tokens to reduce computational cost while maintaining model accuracy. The core of effective token pruning lies in exploring a training-free, lightweight and efficient discriminative criterion to accurately identify semantically critical tokens corresponding to the target classes.

Our token selection scheme is built upon two core assumptions. First, within a single category, the key visual patch tokens representing target objects share highly similar semantics. Second, for different classes within the same deployment subset, their foreground discriminative features follow common patterns and approximate distributions, providing a theoretical basis for fusion into a unified representation.

In ViT, the CLS token continuously aggregates global semantic information through multi-layer self-attention. After iterative feature reinforcement layer by layer within Transformer, the CLS features of images from the same class form stable cluster distributions in the high-dimensional feature space. Accordingly, we take the semantic center of these feature clusters as the class-specific representative vector, which is used to accurately measure the semantic similarity between any image patch token and the target class. Under the constraint of limited support samples, the mean of feature samples serves as the optimal unbiased center estimator that minimizes the intra-class mean square error. Therefore, for each target class c, we take the mean CLS feature of a small number of support samples at pruning layer L as the prototype vector $\mathbf{e}_c \in \mathbb{R}^D$ for this class:

$$\mathbf{e}_c^{(L)} = \frac{1}{N_c} \sum_{i=1}^{N_c} \mathbf{h}_{\text{CLS}}^{(c,i)} \tag{4}$$

where $\mathbf{h}_{\text{CLS}}^{(c,i)} \in \mathbb{R}^D$ denotes the CLS feature of the i-th sample, and N_c is the number of support samples for class c. The construction of class prototypes only requires forward propagation without any parameter update.

During the inference stage, given an input image and the target class c, ViT first divides the image into N patch embeddings with D dimensions, which are concatenated with the CLS token and fed into the Transformer layers. When the feature flow reaches the preset pruning layer L, we extract the feature representation of all image patch tokens at the input of this layer, denoted as $\mathbf{x}_j \in \mathbb{R}^D, (j = 1, \ldots, N)$. To measure the correlation between each token and the target class c, we compute the inner product between $\mathbf{x}_j$ and the prototype vector $\mathbf{e}_c$ as the similarity score:

$$s_j(c) = \mathbf{x}_j^\top \mathbf{e}_c \tag{5}$$

This score reflects the directional consistency between token features and the target class prototype in the feature space; a higher score indicates that the semantics of the token are more consistent with class c. All N tokens are sorted in descending order according to $s_j(c)$, and only the top-k tokens are retained, where $k = \lceil \rho \cdot N \rceil$ and $\rho \in (0, 1]$ is the preset retention ratio. The retained k tokens are concatenated with the CLS token and further fed into layer L and subsequent layers for computation, while the remaining $N - k$ tokens are

directly discarded. The CLS token is not involved in similarity calculation and is always retained unconditionally. Since the pruning operation only involves feature extraction, inner product computation, and sorting without any learnable parameters, CGTS can be directly applied to any pre-trained ViT model without additional fine-tuning.

For multi-class tasks ($K > 1$), we aggregate the CLS feature means of all K classes of support samples at the target layer to construct a global shared prototype vector. During inference, token screening is completed in a single pass relying solely on this single prototype, eliminating repeated forward passes and multi-branch result fusion, thus achieving lightweight and efficient pruning inference. Although different classes within the subset have visual differences, the shared prototype preserves the foreground discriminative semantics common to all target classes and can still effectively capture the overall discriminative semantic region of the target subset. In constrained inference scenarios where the input belongs to one of the K classes, the dot-product operation automatically highlights tokens relevant to the current class. Therefore, the global averaging strategy does not impair token screening effectiveness.

3.3. Integration of MLP Pruning and CGTS

MLP pruning and CGTS adopt a progressive fusion paradigm: we first conduct class-aware structured MLP pruning on target subsets to obtain a low-noise compact backbone and then construct class prototypes based on purified features for CGTS token filtering. The two pruning modules operate on disjoint dimensions, and either standalone strategy bears obvious drawbacks. Sole MLP pruning fails to cut the quadratic computational cost introduced by patch tokens, while standalone CGTS suffers degraded prototype discriminability caused by redundant neurons in the unrefined network. This serial architecture remedies the flaws of both single pruning branches simultaneously, achieving a better balance between classification accuracy and compression efficiency and reducing overall inference latency collaboratively.

4. Evaluation

4.1. Experimental Setup

Data and models:

This paper targets practical edge deployment scenarios that require recognition of a fixed small set of categories. With 100 and 200 classes, respectively, CIFAR-100 and Tiny-ImageNet are well aligned with the category scale of typical real-world edge applications. They also enable us to build multiple nested class subsets to systematically evaluate the performance of our class-aware pruning under varying numbers of target categories.

We use DeiT-Tiny and DeiT-small [2] pre-trained on CIFAR-100 and TinyImageNet as our backbones. The all-classes accuracy of DeiT-Tiny and DeiT-small is 83.79% and 86.18%. We evaluate all methods with four class subsets, $K \in \{5, 10, 20, 50\}$, randomly drawn from the full datasets. All subset results use restricted evaluation : argmax over the K target logits, which aligns with our core deployment premise that all input samples belong to one of the predefined K categories.

It is worth noting that this closed-set restricted evaluation protocol differs from the full-category standard evaluation adopted by general ViT benchmarks. This constrained setup highly matches our targeted edge deployment scenario with fixed target classes and no out-of-distribution inputs in real operation, so we adopt this evaluation rule to precisely fit the inference requirements of the dedicated device studied in this paper.

Implementation details:

During the fine-tuning stage after MLP structured pruning, we adopt the AdamW optimizer with an initial learning rate of 1×10^{-4} and weight decay of 1×10^{-5}. The batch

size is set to 32, and we apply a cosine annealing scheduler without warm-up for a total of five training epochs. For CGTS prototype construction, we randomly draw 10 training images per target class as support samples, and test data is never involved. The index lists of support samples for all subsets are pre-generated offline and loaded directly in inference, which eliminates redundant real-time sampling operations.

Baselines:

We select two types of baselines for comprehensive evaluation: mainstream class-agnostic ViT pruning approaches and the representative class-aware ViT work X-Pruner. We compare against five methods: (1) DynamicViT [18], a learned token predictor inserted at layer 6; (2) SViTE [29], a static class-aware token mask computed at layer 6; (3) X-Pruner [33], class-conditional soft head gating trained with augmented Lagrangian multipliers (ALMs); (4) X-Pruner (hard) , where trained X-Pruner gates are thresholded at 0.5 and heads are structurally removed via weight compaction; and (5) Zero-TPrune [26], a zero-shot token pruning method that scores tokens via Weighted PageRank (WPR) on the self-attention graph, requiring no training or calibration data. All fine-tuned baselines use five epochs on K-class data. Our method additionally applies class-aware Taylor MLP pruning before fine-tuning.

Metrics: We report top-1 accuracy, GMACs per image, backbone size (MB), per-deployment storage overhead, and GPU latency at batch size 64 (NVIDIA Tesla P100, Santa Clara, CA, USA). We run all experiments multiple times with different random seeds for model initialization and random subset sampling and report averaged metrics to guarantee statistical reliability, following standard evaluation protocols for Vision Transformer pruning. Specifically, we used five random class subsets per K and three independent runs per subset.

4.2. Main Results

Table 1 reports the experimental results of DeiT-Tiny evaluated on the CIFAR-100 dataset, covering a comprehensive comparison of classification accuracy and inference efficiency across all methods.

First, compared with the unpruned baseline, our MLP30% + CGTS0.6 method achieves better accuracy on small and medium-scale subsets with $K = 5, 10, 20$ and obtains prominent efficiency gains. Specifically, it reduces GMACs by 31%, cuts model size by 19%, and increases inference throughput by 36% relative to the unpruned baseline.

Further compared with existing mainstream pruning methods, our approach achieves competitive classification accuracy, which is comparable to or even better than most peer schemes. It also delivers consistently superior efficiency: compared with existing pruning methods, our method can further reduce GMACs by an additional 15%, compress model size by 19%, and boost inference throughput by 14%.

Notably, our method achieves slightly lower accuracy than competitors on the large subset $K = 50$. We attribute this performance gap to the single-prototype limitation of CGTS when dealing with heterogeneous large-K subsets.

Meanwhile, X-Pruner with soft gating yields the highest accuracy across all K settings but fails to yield efficiency gains. The extra gating overhead slows down its inference speed by about 9% compared with the unpruned baseline.

By contrast, the hard-gated X-Pruner under the aggressive keep = 0.5 setting suffers catastrophic accuracy degradation, dropping to 83.2% at $K = 20$ and 56.8% at $K = 50$. This phenomenon verifies that soft gating mechanisms cannot be extended to structural model pruning and removal. In addition, under the more aggressive pruning setting of MLP50% + CGTS0.6, the model still maintains stable classification accuracy with only marginal performance decline, while the efficiency gain is further enhanced. Figure 2 intuitively

illustrates the accuracy trends of all compared pruning methods across different target class sizes on the CIFAR-100 dataset.

Table 1. Comprehensive comparison on CIFAR-100 with DeiT-Tiny. † Token keep (physical sequence reduction). ‡ Soft head gating, no FLOPs reduction. * Hard structural head removal after gate thresholding; accuracy varies by K (see text). § Zero-shot WPR scoring; no training or calibration data needed, but adds 8.9% latency overhead and increases peak memory vs CGTS (see Section 4.3). — denotes not applicable.

Method	Train	Accuracy (%)				Efficiency				
		$K = 5$	$K = 10$	$K = 20$	$K = 50$	GMACs	MB	Per-Subset	ms/B64	img/s
Unpruned	—	96.20	95.50	94.35	89.00	1.225	22.17	—	101.6	630
Ours MLP30% + CGTS0.6 †	5ep	97.60	96.10	94.85	88.28	0.844	17.91	0.8 KB	74.8	856
Ours MLP50% + CGTS0.6 †	5ep	96.80	95.40	94.25	87.56	0.636	15.08	0.8 KB	67.8	944
CGTS only † (zero-shot)	none	97.00	95.60	93.95	88.80	1.022	22.17	0.8 KB	83.6	766
Zero-TPrune †,§ (keep = 0.7)	none	96.60	95.80	93.95	88.88	1.030	22.17	0 B	91.1	703
DynamicViT † (keep = 0.7)	5ep × N	97.60	95.20	94.05	89.62	1.022	22.17	22.2 MB	90.3	709
SViTE † h0 (keep = 0.7)	5ep × N	96.20	94.70	92.85	86.86	1.022	22.17	22.2 MB	85.1	751
X-Pruner ‡ ALM (keep = 0.7)	5ep × N	98.60	97.40	95.60	90.10	1.225	22.17	14.1 KB	111.5	574
X-Pruner * hard (keep = 0.7)	5ep × N	97.80	96.60	93.90	87.06	1.225	22.17	14.1 KB	111.5	574
X-Pruner * hard (keep = 0.5)	5ep × N	88.20	94.90	83.15	56.82	≈1.2	19–22	14.1 KB	—	—

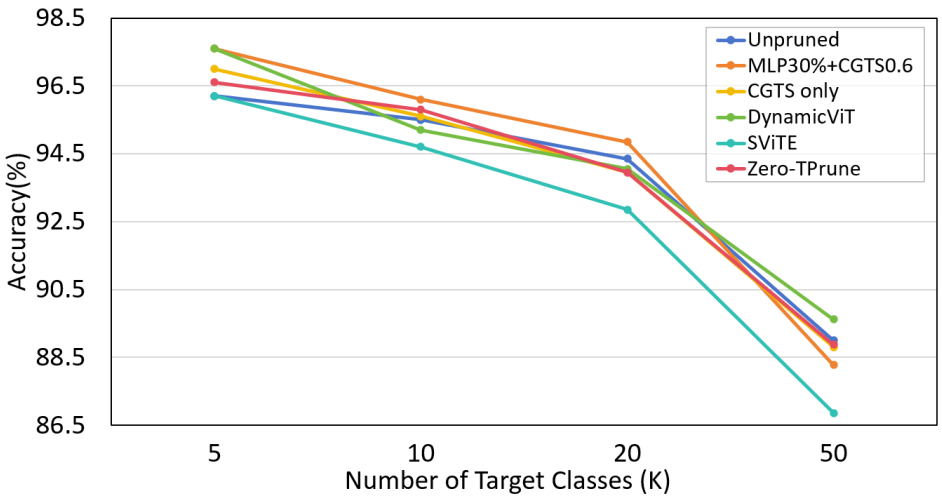


Figure 2. Accuracy of various token/head pruning methods under different numbers of target classes K on CIFAR-100 with DeiT-Tiny ($K = 5, 10, 20, 50$).

In addition, we further conduct generalization validation on the TinyImageNet-200 dataset with high class complexity using the DeiT-Small model, as reported in Table 2. Compared with CIFAR-100, TinyImageNet possesses stronger semantic heterogeneity and higher task difficulty, leading to distinct accuracy trends among various pruning methods. Figure 3 plots the corresponding accuracy curves on the TinyImageNet dataset, showing the performance change rule under a 200-class complex task.

Table 2. Comprehensive comparison on TinyImageNet with DeiT-Small. † Token keep (physical sequence reduction). ‡ Soft head gating, no FLOPs reduction.

Method	$K = 5$	$K = 10$	$K = 20$	$K = 50$
Unpruned (no fine-tune)	99.60	98.20	96.10	94.12
Fine-tuned only (pr = 0.0)	99.60	98.40	95.60	92.92
Ours MLP30% + CGTS0.6 †	98.80	97.80	93.70	90.32
Ours MLP50% + CGTS0.6 †	98.40	96.80	91.80	89.20
DynamicViT † (keep = 0.7)	99.20	97.20	95.20	92.00
SViTE † h0 (keep = 0.7)	98.80	98.00	94.40	92.44
X-Pruner ‡ ALM (keep = 0.7)	99.20	98.60	95.30	93.00

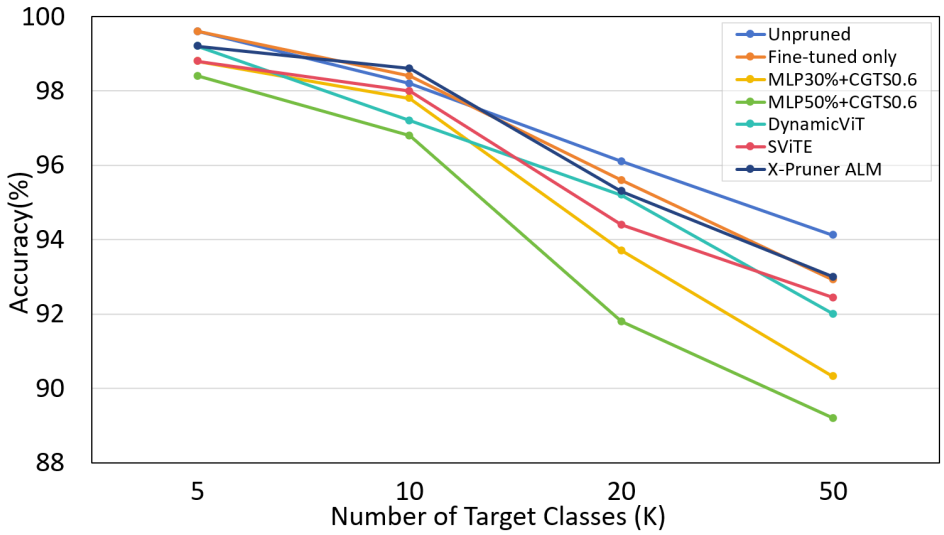


Figure 3. Accuracy of various token/head pruning methods under different numbers of target classes K on TinyImageNet with DeiT-Small ($K = 5, 10, 20, 50$).

Overall, the X-Pruner soft gating scheme achieves the best accuracy across all K settings without physical structural pruning. DynamicViT and SViTE also maintain competitive performance. Although our method delivers slightly lower accuracy than the above competitors, the performance gap is controllable.

At $K = 5$, MLP30% + CGTS0.6 achieves comparable accuracy to SViTE, merely 0.40% lower than DynamicViT. In the cases of $K = 10, 20, 50$, the accuracy difference between our method and mainstream structural pruning methods remains stable within 0.6–1.5%. The core reason lies in the higher complexity of TinyImageNet. The single-prototype mechanism of CGTS suffers more serious representation degradation on heterogeneous large-K subsets. Even so, our method still maintains acceptable classification accuracy, and its core advantage of balancing accuracy and efficiency is well retained.

Overall, compared with the mainstream pruning methods covered in this paper, our approach achieves simultaneous reductions in GMACs, model size, deployment storage cost, and inference latency while maintaining acceptable classification accuracy.

4.3. Efficiency Analysis

Under the batch size setting of $B = 64$, structural MLP compression (physically reducing the dimensions of fc1/fc2) is essential to obtain real latency gains. Soft head gating (X-Pruner) and soft token masking cannot yield actual acceleration under the standard PyTorch 2.5.1 execution paradigm. The token predictor of DynamicViT introduces additional overhead at $B = 64$, resulting in an 8% slower inference speed compared with CGTS under the same GMACs. Zero-TPrune also exhibits considerable extra overhead.

Although Zero-TPrune requires no calibration data with negligible per-subset storage consumption, its WPR scoring mechanism instantiates a complete $[B, H, N, N]$ attention matrix and performs three power iteration steps during each forward propagation. At $B = 64$, its inference latency increases by 8.9% compared with CGTS, and the peak GPU memory rises by 6% from 185 MB of the unpruned baseline to 197 MB. This is because the buffer of $B \times H \times N \times N$ dominates the activation memory consumption. By shortening the sequence length at layers 7–11, CGTS reduces the peak memory by 10%. Our method is the only one that achieves simultaneous reductions across all four efficiency dimensions.

We further analyze the inference scenario of $B = 1$ for edge deployment, with the corresponding results summarized in Table 3. All token pruning methods (CGTS, DynamicViT, and Zero-TPrune) fail to bring efficiency benefits at $B = 1$, due to low GPU utilization and dominant kernel launch overhead. Zero-TPrune performs the worst at $B = 1$, with inference latency increased by 4% compared with the baseline, verifying that its per-sample WPR iteration cost becomes a notable burden for edge deployment. In contrast, structural MLP compression remains effective and achieves a 16% speedup. Even under low arithmetic intensity conditions, smaller weight matrices effectively reduce memory bandwidth consumption. This further demonstrates the deployment advantage of our proposed method.

Table 3. Batch size B = 1 latency (edge deployment scenario).

Method	B = 1 (ms)	vs. Unpruned
Unpruned	7.18	—
Ours MLP30% + CGTS0.7	6.19	+16%
Ours MLP50% + CGTS0.5	6.82	+5%
CGTS only (keep = 0.7)	7.29	−2%
CGTS only (keep = 0.5)	6.97	+3%
Zero-TPrune (keep = 0.7)	7.45	−4%
Zero-TPrune (keep = 0.5)	7.20	0%
DynamicViT (keep = 0.7)	7.16	0%
SViTE (keep = 0.7)	5.39	+33% †
X-Pruner (keep = 0.7)	7.34	−2%

† Static precomputed index gather; requires 22.2 MB checkpoint per deployment and delivers lower accuracy (−1.5% at $K = 20$).

4.4. Memory Overhead Analysis

CGTS only needs to store a 768-byte prototype vector for the deployment of each class subset, without retaining a separate complete model. In contrast, both DynamicViT and SViTE require saving an independent full model checkpoint for each class subset. When the number of deployed subsets is $N = 100$, the overall storage overhead of our method is only 22.2 MB (backbone + 100 × 0.8 KB), which is 1/100 of that of DynamicViT (2.22 GB). In addition, the deployment of a new K-class subset only requires one lightweight forward pass over support samples to generate prototypes, with no retuning required; prototypes only need recalculation when the full target category set is replaced. For multiple independent target subsets, our shared fixed backbone incurs negligible extra storage and computation overhead, showing remarkable advantages in deployment efficiency and rapid iteration.

4.5. Ablation Study

In this section, we conduct ablation experiments on the TinyImageNet dataset (Table 4). Taking the fine-tuned model without pruning and CGTS as the baseline, standalone MLP pruning with a ratio of 30% yields an accuracy drop below 0.5% at $K = 5, 10$, achieving

near-lossless compression. For standalone CGTS with a keep ratio of 0.6, accuracy degradation increases with K, rising from 0.80% at $K = 5$ to 0.96% at $K = 50$, which verifies the representation degradation law of the single-prototype mechanism. The losses are approximately additive when the two modules are combined. Taking $K = 50$ as an example, the accuracy loss of MLP30% + CGTS0.6 is 2.60%, slightly higher than the summed individual loss of 1.88%, while no abrupt performance collapse is observed, demonstrating satisfactory compatibility. Under the aggressive configuration of MLP50% + CGTS0.6, the accuracy reaches 89.20% at $K = 50$, with a drop of 3.72% compared with the baseline. More importantly, joint deployment can compress both MLP width and sequence length simultaneously, achieving multiplicative efficiency gains unavailable with any single module (FLOPs reduced by 31%, and throughput improved by nearly 50%). It is validated that the collaboration of the two modules can achieve prominent efficiency improvement at the cost of acceptable accuracy loss, thus enabling a flexible trade-off between model accuracy and compression ratio.

Table 4. Ablation study of our components on TinyImageNet (DeiT-Small, 5 fine-tuning epochs, CGTS at layer 6).

Method	$K = 5$	$K = 10$	$K = 20$	$K = 50$
Unpruned (Fine-tuned)	99.60	98.40	95.60	92.92
Only MLP Prune 30%	99.20	98.40	94.30	92.00
Only MLP Prune 50%	99.20	97.60	92.70	91.04
Only CGTS (keep = 0.6)	98.80	97.80	94.70	91.96
MLP30% + CGTS0.6	98.80	97.80	93.70	90.32
MLP50% + CGTS0.6	98.40	96.80	91.80	89.20

5. Conclusions

This paper proposes a class-aware joint pruning framework for Vision Transformers, which performs structured MLP pruning and dynamic token pruning simultaneously to tailor pre-trained ViT models into dedicated models for target class subsets. Extensive experiments demonstrate that the proposed method can reduce GMACs by up to 48% and improve inference throughput by nearly 50% while maintaining high accuracy on target classes, with only 0.8 KB storage overhead per single class. It outperforms several mainstream pruning methods in the accuracy–efficiency trade-off and enables flexible multi-subset deployment with a single backbone network.

Nevertheless, several directions remain for optimizing our framework in more complex real-world scenarios. Our single shared prototype is built under the assumption of consistent feature patterns within a class subset, which may lead to performance degradation when facing highly heterogeneous categories. Moreover, the current prototype construction pipeline leverages clean, identically distributed support samples; future work could further boost token pruning robustness to handle noisy labels and cross-domain distribution mismatches. Additionally, our method has substantial potential to deliver better compression gains when tailored for fine-grained classification tasks with subtle inter-class differences.

In the future, we aim to improve the single-prototype mechanism of CGTS by introducing multi-prototype learning or adaptive prototype representation strategies to mitigate feature degradation of the model on large-scale and highly heterogeneous class subsets and further reduce the accuracy gap in complex scenarios. Meanwhile, we can combine joint pruning with quantization and knowledge distillation to adapt to edge devices and extend the class-aware pruning to more ViT variants and high-level vision tasks as well as large-scale benchmarks including ImageNet-1K and COCO to verify the universality

and scalability of the method, providing a more comprehensive technical solution for lightweight Transformer deployment in practical edge scenarios.

Author Contributions: Conceptualization, Z.L. and T.L.; Methodology, Z.L. and T.L.; Software, Z.L.; Validation, F.H., X.H. and Z.Z.; Formal analysis, Z.L.; Investigation, Z.L. and T.L.; Resources, T.L.; Data curation, F.H. and X.H.; Visualization, Z.Z.; Supervision, T.L.; Project administration, T.L.; Funding acquisition, T.L.; Writing—original draft, Z.L.; Writing—review & editing, T.L. and F.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Higher Education Talent Training Quality and Teaching Reform Projects of Chengdu University of Technology under Grant JG2430165 and JG2420017.

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable

Data Availability Statement: The data presented in this study are available in public open-source datasets CIFAR-100 and TinyImageNet-200. These data were derived from the following resources available in the public domain: CIFAR-100 (https://www.cs.toronto.edu/~kriz/cifar.html, accessed on 2 August 2026) and TinyImageNet-200 (http://cs231n.stanford.edu/tiny-imagenet-200.zip, accessed on 2 August 2026).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929 .
2. Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; Jégou, H. Training data-efficient image transformers & distillation through attention. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021; pp. 10347–10357.
3. Yuan, L.; Chen, Y.; Wang, T.; Yu, W.; Shi, Y.; Jiang, Z.H.; Tay, F.E.; Feng, J.; Yan, S. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 558–567.
4. Graham, B.; El-Nouby, A.; Touvron, H.; Stock, P.; Joulin, A.; Jégou, H.; Douze, M. Levit: A vision transformer in convnet's clothing for faster inference. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 12259–12269.
5. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-end object detection with transformers. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; pp. 213–229.
6. Chen, J.; Shu, Q.; Lu, Y.; Zhang, Y.; Wang, Y. QCTF: A Quantized Communication and Transferable Fusion Framework for Multi-Agent Collaborative Perception. *IEEE Trans. Intell. Transp. Syst.* **2025**, *26*, 15013–15027. [CrossRef]
7. Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; Dai, J. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv* **2020**, arXiv:2010.04159.
8. Fang, Y.; Liao, B.; Wang, X.; Fang, J.; Qi, J.; Wu, R.; Niu, J.; Liu, W. You only look at one sequence: Rethinking transformer in vision through object detection. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 26183–26197.
9. Xie, E.; Wang, W.; Yu, Z.; Anandkumar, A.; Alvarez, J.M.; Luo, P. SegFormer: Simple and efficient design for semantic segmentation with transformers. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 12077–12090.
10. Zheng, S.; Lu, J.; Zhao, H.; Zhu, X.; Luo, Z.; Wang, Y.; Fu, Y.; Feng, J.; Xiang, T.; Torr, P.H.; et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Virtual, 19–25 June 2021; pp. 6881–6890.
11. Gu, J.; Kwon, H.; Wang, D.; Ye, W.; Li, M.; Chen, Y.H.; Lai, L.; Chandra, V.; Pan, D.Z. Multi-scale high-resolution vision transformer for semantic segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 12094–12103.
12. Luo, X.; Liu, D.; Kong, H.; Huai, S.; Chen, H.; Xiong, G.; Liu, W. Efficient deep learning infrastructures for embedded computing systems: A comprehensive survey and future envision. *ACM Trans. Embed. Comput. Syst.* **2024**, *24*, 1–100. [CrossRef]
13. Cheng, H.; Zhang, M.; Shi, J.Q. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 10558–10578. [CrossRef] [PubMed]
14. Liu, Z.; Sun, M.; Zhou, T.; Huang, G.; Darrell, T. Rethinking the value of network pruning. *arXiv* **2018**, arXiv:1810.05270.

15. Yang, H.; Yin, H.; Shen, M.; Molchanov, P.; Li, H.; Kautz, J. Global vision transformer pruning with hessian-aware saliency. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 18–22 June 2023; pp. 18547–18557.
16. Zheng, C.; Zhang, K.; Yang, Z.; Tan, W.; Xiao, J.; Ren, Y.; Pu, S. Savit: Structure-aware vision transformer pruning via collaborative optimization. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 9010–9023. [CrossRef]
17. Xu, K.; Wang, Z.; Chen, C.; Geng, X.; Lin, J.; Yang, X.; Wu, M.; Li, X.; Lin, W. Lpvit: Low-power semi-structured pruning for vision transformers. In Proceedings of the European Conference on Computer Vision, Milan, Italy, 29 September–4 October 2024; pp. 269–287.
18. Rao, Y.; Zhao, W.; Liu, B.; Lu, J.; Zhou, J.; Hsieh, C.J. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 13937–13949.
19. Kong, Z.; Dong, P.; Ma, X.; Meng, X.; Niu, W.; Sun, M.; Shen, X.; Yuan, G.; Ren, B.; Tang, H.; et al. Spvit: Enabling faster vision transformers via latency-aware soft token pruning. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–27 October 2022; pp. 620–640.
20. Liang, Y.; Ge, C.; Tong, Z.; Song, Y.; Wang, J.; Xie, P. Not all patches are what you need: Expediting vision transformers via token reorganizations. *arXiv* **2022**, arXiv:2202.07800.
21. Qin, Z.; Yu, F.; Liu, C.; Chen, X. CAPTOR: A class adaptive filter pruning framework for convolutional neural networks in mobile applications. In Proceedings of the 24th Asia and South Pacific Design Automation Conference, Tokyo, Japan, 21–24 January 2019; pp. 444–449.
22. Aggarwal, S.; Binici, K.; Mitra, T. CRISP: Hybrid structured sparsity for class-aware model pruning. In Proceedings of the 2024 Design, Automation & Test in Europe Conference & Exhibition (DATE), Valencia, Spain, 25–27 March 2024; pp. 1–6.
23. Ma, Y.D.; Zhao, Z.C.; Liu, D.; He, Z.; Zhou, W. OCAP: On-device Class-Aware Pruning for personalized edge DNN models. *J. Syst. Archit.* **2023**, *142*, 102956. [CrossRef]
24. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
25. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
26. Wang, H.; Dedhia, B.; Jha, N.K. Zero-tprune: Zero-shot token pruning through leveraging of the attention graph in pre-trained transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 17–21 June 2024; pp. 16070–16079.
27. Shen, C. Adaptive MLP Pruning for Large Vision Transformers. *arXiv* **2026**, arXiv:2603.08100.
28. Shen, C.; Zhu, H.; Fang, G.; Wang, J.; Wang, X. Diversity-Guided MLP Reduction for Efficient Large Vision Transformers. *arXiv* **2025**, arXiv:2506.08591.
29. Chen, T.; Cheng, Y.; Gan, Z.; Yuan, L.; Zhang, L.; Wang, Z. Chasing sparsity in vision transformers: An end-to-end exploration. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 19974–19988.
30. Hemmat, M.; Miguel, J.S.; Davoodi, A. Cap'nn: A class-aware framework for personalized neural network inference. *ACM Trans. Embed. Comput. Syst.* **2022**, *21*, 1–24. [CrossRef]
31. Hu, W.; Che, Z.; Liu, N.; Li, M.; Tang, J.; Zhang, C.; Wang, J. CATRO: Channel pruning via class-aware trace ratio optimization. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *35*, 11595–11607. [CrossRef] [PubMed]
32. Zhang, H.; Liu, L.; Huang, Y.; Yang, Z.; Lei, X.; Wen, B. Cakdp: Category-aware knowledge distillation and pruning framework for lightweight 3d object detection. In Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 17–21 June 2024; pp. 15331–15341.
33. Yu, L.; Xiang, W. X-pruner: Explainable pruning for vision transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 18–24 June 2023; pp. 24355–24363.
34. Wei, Z.; He, Q.; Chen, F.; Duan, R.; Li, X.; Li, B.; Chen, Y.; Xue, H.; Jin, H.; Yang, Y. Vulcan: Crafting Compact Class-Specific Vision Transformers For Edge Intelligence. In Proceedings of the The Fourteenth International Conference on Learning Representations, Rio de Janeiro, Brazil, 23–27 April 2026.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.