CLASS-SPECIFIC PRUNING OF VISION TRANSFORMERS VIA ACTIVATION CALIBRATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing class-specific pruning methods compress models for target classes but mainly evaluate neuron importance using activation magnitudes, treating all tokens equally. Our analysis reveals that tokens with high activation responses are not concentrated on target foreground objects; instead, many lie in background regions unrelated to the target classes, interfering with the identification of category-critical neurons. Further analysis shows using frequency stability and attention weights to calibrate the activation contributions of different tokens can mitigate interference from irrelevant background information in neuron importance evaluation. Based on these findings, we propose Activation-Calibrated Pruning (AC-Prune), which identifies category-critical neurons based on the calibrated activation contributions. The selected neurons then serve as anchors to guide feed-forward network (FFN) neuron consolidation, while low-rankness is encouraged in multi-head attention (MHA) weights. This mitigates irreversible knowledge loss caused by direct pruning, yielding compact and efficient class-specific Vision Transformers (ViTs). Comprehensive experiments with five base ViTs covering three representative visual tasks on four datasets demonstrate that AC-Prune-derived ViTs retain approximately 20%–40% of the base models' parameters while outperforming them on class-specific tasks by up to 15.72% in accuracy.

1 INTRODUCTION

Vision Transformers (ViTs) have achieved remarkable success across a variety of vision tasks, including image recognition (Fixelle, 2025; Lu et al., 2025), object detection (Wang et al., 2025a; Yu et al., 2024), and instance segmentation (Jain et al., 2023; Ravi et al., 2025). Their progress is largely driven by the continuous scaling of model size, which enhances representational capacity and generalization performance (Wang et al., 2024a; 2025b; Han et al., 2025). However, while scaling pushes the frontiers of performance, it incurs high computational costs and significant memory footprints (Saha & Xu, 2025). Edge deployment scenarios, such as intelligent vehicles, impose requirements for real-time processing and privacy preservation, highlighting the urgent need to unlock ViT's potential on edge devices through model compression (Tuli & Jha, 2024; Ye et al., 2024).

To address this deployment challenge, numerous model compression methods have been proposed to effectively adapt ViTs to edge devices (Ye et al., 2024; Liao et al., 2023; Tuli & Jha, 2024). Among these methods, structured pruning (Cheng et al., 2024) is particularly suitable for edge deployment due to its strong hardware compatibility and scalability. However, as illustrated in Fig. 1(a), existing structured pruning methods (Shen et al., 2025; Zhang et al., 2024; Yu & Xiang, 2023; Yang et al., 2023) typically aim to preserve the overall performance across all classes, compressing all categories uniformly. This objective does not fully align with practical edge requirements: models typically need only category-specific knowledge rather than the full class knowledge embedded in large-scale pre-trained models (Liao et al., 2023; Sun et al., 2025; Cao et al., 2025). For instance, in autonomous driving, the model only needs to recognize traffic-relevant categories such as vehicles, pedestrians, and traffic signs, while it is unnecessary to identify irrelevant classes like furniture or flowers. Such irrelevant knowledge can interfere with the model's focus on target categories, leading to noticeable performance degradation (Geirhos et al., 2020). Therefore, compressing a generic model across all classes alone is insufficient to meet this practical requirement; compact and accurate class-specific models must instead be derived from generic pre-trained ViTs.

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.

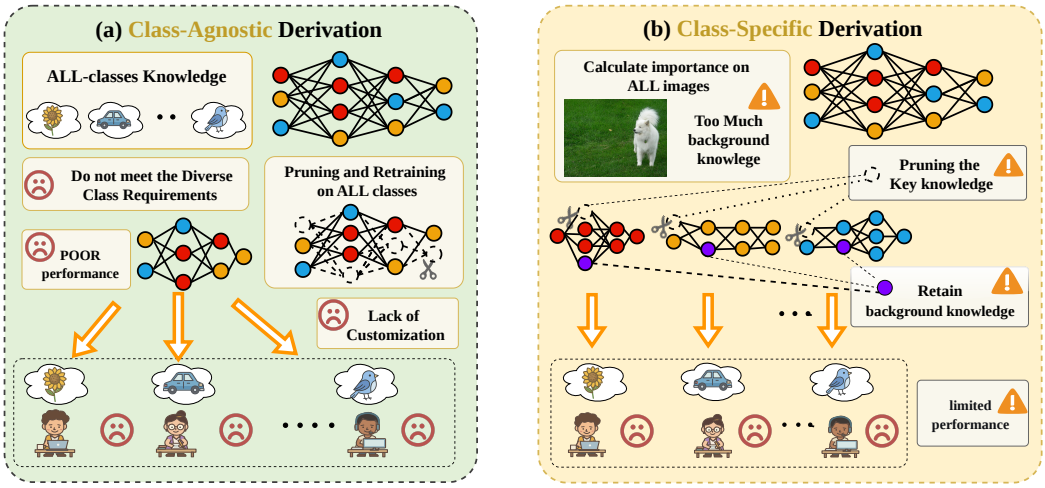


Figure 1: Limitations of existing pruning methods: (a) class-agnostic pruning lacks class-specific customization; (b) existing class-specific pruning retains irrelevant background knowledge.

Prior studies have found that FFNs in ViTs primarily encode class-specific knowledge, whereas MHAs predominantly capture class-agnostic patterns, and have explored class-specific pruning based on this observation (Wei et al., 2026b;a). However, as illustrated in Fig. 1(b), existing class-specific pruning methods primarily assess neuron importance using activation magnitudes, overlooking differences in the contributions of individual tokens to the target classes. Background tokens can produce strong activation responses. As shown in Fig. 2, high-response tokens selected by activation magnitude are not always concentrated on target objects and can also occur in background regions. Such background responses can interfere with the selection of category-critical neurons and impair the pruned model's performance on the target task. To quantify this phenomenon, we compare activation-based and uniform random selection of 20% of patch tokens across nine ImageNet-1K sub-tasks constructed by randomly sampling different numbers of classes. As shown in Fig. 3, activation-based selection yields a lower inside-box proportion than random selection at Layers 8 and 12 on all nine sub-tasks. These results indicate that larger activation magnitudes do not necessarily imply greater relevance to the target foreground. Thus, evaluating neuron importance directly from activation magnitudes can be affected by irrelevant background information.



Figure 2: Tokens with the highest activation magnitudes (top 20%) are highlighted in blue.

To reduce interference from irrelevant background information in neuron selection, inspired by prior work (Shi et al., 2026), we use frequency stability to calibrate the activation contributions of different tokens. As shown in Fig. 4, tokens selected after frequency-based calibration are more concentrated on the target object than those selected by activation magnitude alone. Building on this, we further use attention weights to calibrate token activation contributions and identify category-critical neurons. As shown in Fig. 5, across the nine sub-tasks, the top 20% of tokens selected based on these calibrated contributions achieve a higher average inside-box proportion than those selected using activation magnitude alone or frequency-based calibration alone.

Based on the above analysis, we propose Activation-Calibrated Pruning (AC-Prune) to derive compact and efficient class-specific ViTs from a pre-trained model. AC-Prune identifies category-critical neurons based on calibrated activation contributions and performs Anchor-Centric Neuron Consolidation (ACNC) using these neurons as anchors, while encouraging low-rankness in MHA weights through Truncated Nuclear Norm Regularization (TNNR). This process mitigates knowledge loss

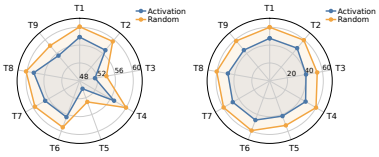


Figure 3: Inside-box proportions at Layers 8 (left) and 12 (right): activation vs. random sampling.



Figure 4: Token selection without (left) and with (right) frequency guidance.

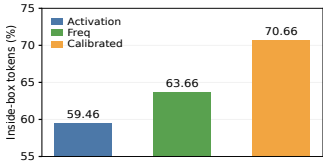


Figure 5: Average inside-box proportions for Activation, Freq, and Calibrated.

caused by direct pruning, after which structured pruning yields compact models tailored to the target classes. An overview is shown in Fig. 6. Our main contributions are summarized as follows:

- We analyze the limitations of evaluating neuron importance solely using activation magnitudes, revealing that the high activation responses from irrelevant background tokens can substantially interfere with the accurate identification of category-critical neurons, and we mitigate this interference through calibration with frequency stability and attention weights.
- We propose AC-Prune, which selects anchor neurons based on calibrated activation contributions to guide FFN neuron consolidation and encourages low-rankness in MHA weights, preparing the model for compression before pruning, which minimizes knowledge loss during pruning.
- Extensive experiments with five base ViT models covering three visual tasks and four benchmark datasets demonstrate that AC-Prune substantially reduces model size while outperforming the base models and existing state-of-the-art structured pruning methods on class-specific tasks.

2 Background and Related Work

2.1 Edge Model Deployment

Recently, a series of increasingly large ViTs have been developed (Wang et al., 2024a; 2025b; Han et al., 2025), alongside a rapidly growing demand for deploying such models on resource-constrained edge devices to enable real-time responsiveness, privacy preservation, and reliable service. To address this tension, edge-friendly architectures such as EfficientViT (Liu et al., 2023), FLatten Transformer (Han et al., 2023), and HSViT (Xu et al., 2025) have been proposed. While effective, these approaches rely on costly manual architecture design, require training from scratch, and struggle to keep pace with the rapid evolution of large ViTs, limiting their ability to deliver increasingly powerful models for diverse edge scenarios. Different from these efforts, our AC-Prune derives class-specific compact ViTs from a pre-trained base ViT tailored to target edge domains, enabling the edge model to inherit the knowledge of the base model while focusing on the features essential to target classes and avoiding interference from irrelevant category knowledge, thereby achieving efficient and highly task-oriented model development.

2.2 Model Compression

Various compression methods enable edge deployment of ViTs, including quantization (Choi & Kim, 2025; Zhong et al., 2025), knowledge distillation (Zhang et al., 2022; Cao et al., 2025), as well as structured (Sun et al., 2025; Zhang et al., 2024) and unstructured pruning (Chen et al., 2021; Gupta et al., 2024). However, not all are edge-friendly. Quantization and unstructured pruning typically rely on specialized infrastructure for acceleration, which limits their applicability on diverse edge devices (Yang et al., 2024; Cheng et al., 2024). Knowledge distillation transfers knowledge only from the feature space, incurring substantial training cost (He et al., 2025). In contrast, structured pruning extracts knowledge from the parameter space and yields regularly shaped models that can be deployed across diverse edge devices, which serves as the foundation of AC-Prune. In comparison, AC-Prune targets a specific subset of classes and uses activation calibration to mitigate interference from irrelevant background information in the identification of category-critical neurons. Building on this, AC-Prune follows a train-then-prune paradigm that aggregates knowledge before weight removal to mitigate irreversible knowledge loss and enable near-lossless pruning.

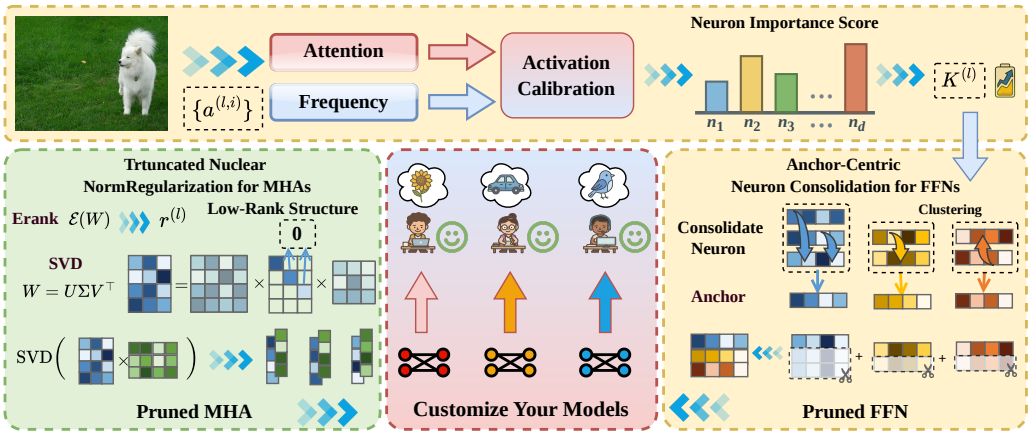


Figure 6: Overview of AC-Prune. Activation calibration uses frequency stability and attention weights to calibrate token contributions and identify anchor neurons. ACNC consolidates FFN neurons around these anchors, while TNNR induces low-rank structure in MHA weights. Structured pruning then produces compact and efficient class-specific ViTs tailored to the target tasks.

3 AC-Prune: Class-Specific Model Derivation

3.1 Notation and Preliminaries

Vision Transformer. A ViT (Dosovitskiy et al., 2021) is composed of multiple Transformer blocks, each containing an MHA module and an FFN module. Given a sequence of patch tokens $X \in \mathbb{R}^{N\times d}$, where N is the number of tokens and d is the embedding dimension, each MHA head computes and aggregates contextual representations through attention-weighted aggregation of token features:

$$\mathrm{MHA}^{(l)}(X) = \sum_{h=1}^{H_l} \mathrm{Attn}^{(l,h)}(X) = \sum_{h=1}^{H_l} \mathrm{softmax}\left(\frac{XW_Q^{(l,h)\top} W_K^{(l,h)} X^\top}{\sqrt{q_l}}\right) XW_V^{(l,h)\top} W_O^{(l,h)\top}, \tag{1}$$

where $W_{Q|K}^{(l,h)} \in \mathbb{R}^{q_l\times d}$, $W_V^{(l,h)} \in \mathbb{R}^{v_l\times d}$ and $W_O^{(l,h)} \in \mathbb{R}^{d\times v_l}$ are the query, key, value, and output projection matrices for the h-th head in the l-th block. Bias terms are omitted for brevity.

Following MHA, the FFN module applies a two-layer Multi-Layer Perceptron (MLP):

$$\mathrm{FFN}^{(l)}(X) = \sigma(XW_1^{(l)\top})W_2^{(l)\top} = \sum_{i=1}^{e_l} \sigma(XW_1^{(l)}[i]) \otimes W_2^{(l)\top}[i] = \sum_{i=1}^{e_l} \sigma(Xn_1^{(l,i)}) \otimes n_2^{(l,i)}, \tag{2}$$

where $W_1^{(l)} \in \mathbb{R}^{e_l\times d}$ and $W_2^{(l)} \in \mathbb{R}^{d\times e_l}$ are the FFN projection matrices in l-th block, $\sigma(\cdot)$ is a nonlinear activation (e.g., GELU), $n_{1|2}^{(l,i)} \in \mathbb{R}^d$ is the i-th neuron in $W_{1|2}^{(l)}$, and $\otimes$ is the outer product.

Sub-Task. Let M_B be the pre-trained base ViT trained on D_Y for classes $Y = \{y_1, \ldots, y_{|Y|}\}$. AC-Prune aims to derive a compact class-specific edge ViT M_E from M_B, specialized for a subset $S \subseteq Y$. We refer to S as a sub-task of M_B, and denote its corresponding dataset by D_S.

3.2 Activation-Calibrated Neuron Importance

Frequency Stability Score. Foreground token features tend to exhibit smaller variations along the channel dimension than background features (Shi et al., 2026). We therefore use the stability of token features under channel-wise low-pass filtering to calibrate their activation contributions. Given the MLP activations $\mathbf{X} \in \mathbb{R}^{N\times D}$, we apply a one-dimensional Fourier transform along the feature dimension of each token, followed by Gaussian low-pass filtering and inverse transformation:

$$\mathbf{X}_{\mathrm{FFT}} = \mathrm{FFT}_{1\mathrm{D}}(\mathbf{X}), \qquad \mathbf{X}_{\mathrm{LP}} = \mathbf{X}_{\mathrm{FFT}} \odot \mathbf{G},$$
$$\hat{\mathbf{X}} = \Re\{\mathrm{IFFT}_{1\mathrm{D}}(\mathbf{X}_{\mathrm{LP}})\}. \tag{3}$$

Here, $\mathbf{G} \in [0,1]^{N\times D}$ contains the same Gaussian low-pass filter weights applied to all tokens, $\odot$ denotes the element-wise multiplication, and $\Re\{\cdot\}$ extracts the real part. The frequency stability score for each individual token-feature pair is then defined as:

$$\mathbf{S} = \frac{\hat{\mathbf{X}}}{|\hat{\mathbf{X}} - \mathbf{X}| + \epsilon}. \tag{4}$$

A larger value in $\mathbf{S}$ indicates a stronger low-pass response relative to the reconstruction error, suggesting greater stability under low-pass filtering. For each feature dimension, we select the top-K tokens with the largest frequency-stability values in $\mathbf{S}$, where $K = \lfloor N \cdot \alpha \rfloor$ and α denotes the selection ratio. Let $\mathcal{I}_K(j)$ denote the index set of the top-K tokens in the j-th feature dimension. The frequency stability score of token i is defined as its normalized vote count:

$$v_i = \frac{1}{KD} \sum_{j=1}^{D} \mathbb{I}[i \in \mathcal{I}_K(j)], \qquad i = 1, \ldots, N, \tag{5}$$

where $\mathbb{I}[\cdot]$ is the indicator function and D denotes the token feature dimension. A higher v_i indicates greater low-pass stability and thus greater foreground likelihood.

Neuron Importance Score. For architectures with an explicit class token (e.g., DeiT), we use the CLS attention $A_{\text{cls},i}$ and frequency stability v_i to calibrate the activation contributions of patch tokens, then combine these calibrated contributions with the class-token activation to compute neuron importance. The importance of the c-th neuron is defined as:

$$I_c = \lambda_{\text{cls}} \cdot |x_{\text{cls}}(c)| + \lambda_{\text{patch}} \cdot \frac{1}{|\mathcal{P}|} \sum_{i\in\mathcal{P}} A_{\text{cls},i} \cdot v_i \cdot |x_i(c)|, \tag{6}$$

where $\mathcal{P}$ denotes the set of all patch-token indices, $|x_{\text{cls}}(c)|$ is the activation magnitude of the class token on the c-th neuron, and λ_{cls} and λ_{patch} control the importance weights of the class token and patch tokens, respectively (e.g., $\lambda_{\text{cls}} = 2$, $\lambda_{\text{patch}} = 1$). For architectures without an explicit class token (e.g., Swin Transformer), the importance of the c-th neuron is defined as:

$$I_c = \frac{1}{N} \sum_{n=1}^{N} \omega_n \cdot v_n \cdot |x_n(c)|, \tag{7}$$

where $\omega_n = \sum_m \text{mean}_h A_{h,m,n}$ denotes the attentiveness of the n-th token, i.e., the sum of average attention weights from other tokens m to the target token n across all attention heads. We use the frequency stability v_n and attentiveness ω_n to calibrate each token's activation contribution and average the calibrated contributions to obtain neuron importance. The final importance scores I_c serve as the pruning criterion, retaining high-importance neurons.

3.3 Anchor-Centric Neuron Consolidation for FFNs

For a sub-task S, AC-Prune computes the neuron importance score $I_i^{(l)}$ for each FFN neuron using D_S (see §3.2). Since neurons in different blocks specialize in different functions, AC-Prune adaptively determines the number of clusters per block based on the global importance-score distribution and the overall pruning rate $R \in (0,1)$. Let $\mathcal{I} = \bigcup_{l=1}^{L} \{I_i^{(l)} \mid i = 1, \ldots, e_l\}$ collect the global importance scores. The threshold and cluster counts are given as follows:

$$\tau = \Phi\left(\mathcal{I}, \left\lceil R \sum_{j=1}^{L} e_j \right\rceil\right), \qquad K^{(l)} = \sum_{i=1}^{e_l} \mathbb{I}\left[I_i^{(l)} > \tau\right]. \tag{8}$$

Here $\Phi(\mathcal{I}, k)$ returns the k-th smallest element in $\mathcal{I}$ and $\mathbb{I}[\cdot]$ is the indicator function, allowing a single threshold to allocate different retained dimensions across blocks. With $K^{(l)}$ determined, AC-Prune performs k-means clustering on FFN neurons within each block based on their weights to obtain clusters $C_k^{(l)}$. The neuron with the highest importance score in each cluster is selected as its anchor $\hat{n}_k^{(l)}$ to guide the subsequent consolidation of neurons within that cluster. We then perform

ACNC, encouraging the weights of neurons within each cluster to approach those of the corresponding anchor. This is accomplished by introducing a consolidation regularization term during training:

$$\mathcal{L}_{\text{collapse}} = \sum_{l=1}^{L} \frac{1}{K^{(l)}} \sum_{k=1}^{K^{(l)}} \sum_{i=1}^{|C_k^{(l)}|} \left\| \mathbf{W}_k^{(l,i)} - \hat{\mathbf{W}}_k^{(l)} \right\|, \tag{9}$$

where $\mathbf{W}_k^{(l,i)}$ denotes the weight of the i-th neuron in cluster $C_k^{(l)}$, and $\hat{\mathbf{W}}_k^{(l)}$ is the weight of its highest-scoring anchor defined above. This regularization consolidates neurons toward category-critical anchors, preserving class-specific knowledge while enabling each cluster to be represented by a single anchor neuron for subsequent structured compression of the FFN modules.

3.4 Truncated Nuclear Norm Regularization for MHAs

To facilitate subsequent MHA compression via singular value decomposition (SVD), we use TNNR to induce pruning-friendly low-rank structure in the query and value projection matrices. Given the overall pruning rate R, we allocate pruning ratios to the QK and VO dimensions using their layer-wise effective ranks:

$$R_{QK|VO}^{(l)} = \frac{2r_{VO|QK}^{(l)}}{r_{QK}^{(l)} + r_{VO}^{(l)}} R^{(l)}, \qquad R^{(l)} = \frac{L}{\sum_{j=1}^{L} \frac{1}{r_{QK}^{(j)} + r_{VO}^{(j)}}} \cdot \frac{R}{r_{QK}^{(l)} + r_{VO}^{(l)}}, \tag{10}$$

$$r_{QK|VO}^{(l)} = \sum_{h=1}^{H_l} E(W_{QK|VO}^{(l,h)}), \qquad E(W) = \exp\left(-\sum_i p_i \log p_i \right), \qquad p_i = \frac{\sigma_i}{\sum_j \sigma_j}, \tag{11}$$

where the compact notation $R_{QK|VO}^{(l)}$ uses the opposite effective rank in its numerator: $R_{QK}^{(l)}$ uses $r_{VO}^{(l)}$, whereas $R_{VO}^{(l)}$ uses $r_{QK}^{(l)}$. Moreover, $W_{QK}^{(l,h)} = W_Q^{(l,h)\top} W_K^{(l,h)}$, $W_{VO}^{(l,h)} = W_V^{(l,h)\top} W_O^{(l,h)\top}$, $E(W)$ is the effective rank of matrix W, and σ_i is its i-th singular value. Using the allocated pruning ratios, we define the regularization term on the query and value projection matrices as follows:

$$\mathcal{L}_{\text{rank}} = \sum_{l=1}^{L} \sum_{h=1}^{H_l} \left(\sum_{i=q_l'+1}^{q_l} \sigma_Q^{(l,h,i)} + \sum_{i=v_l'+1}^{v_l} \sigma_V^{(l,h,i)} \right), \tag{12}$$

where the retained dimensions are $q_l' = \lfloor q_l(1 - R_{QK}^{(l)}) \rfloor$ and $v_l' = \lfloor v_l(1 - R_{VO}^{(l)}) \rfloor$, with q_l and v_l denoting the original query and value dimensions. This regularization pulls the tail singular values toward zero, inducing low-rank structure before pruning.

3.5 Constrained Optimization and Model Pruning

Constrained Optimization. We combine ACNC and TNNR, defined in Eqs. (9) and (12), respectively, through an augmented Lagrangian objective to jointly regularize FFN and MHA modules, using linear and quadratic penalty terms with learnable multipliers λ_1 and λ_2:

$$\begin{aligned} \mathcal{L} = \mathcal{L}_{\text{task}} &+ \sum_{l=1}^{L} \frac{1}{K^{(l)}} \sum_{k=1}^{K^{(l)}} \sum_{i=1}^{|C_k^{(l)}|} \left(\lambda_1 \left\| \mathbf{W}_k^{(l,i)} - \hat{\mathbf{W}}_k^{(l)} \right\| + \lambda_2 \left\| \mathbf{W}_k^{(l,i)} - \hat{\mathbf{W}}_k^{(l)} \right\|^2 \right) \\ &+ \sum_{l=1}^{L} \sum_{h=1}^{H_l} \left[\sum_{i=q_l'+1}^{q_l} \left(\lambda_1 \sigma_Q^{(l,h,i)} + \lambda_2 \left(\sigma_Q^{(l,h,i)} \right)^2 \right) + \sum_{i=v_l'+1}^{v_l} \left(\lambda_1 \sigma_V^{(l,h,i)} + \lambda_2 \left(\sigma_V^{(l,h,i)} \right)^2 \right) \right], \end{aligned} \tag{13}$$

where $\mathcal{L}_{\text{task}}$ is the task-specific loss, and λ_1, λ_2 are globally shared Lagrange multipliers initialized to zero and updated during training by gradient ascent with learning rate ρ.

Model Pruning. After constrained training, AC-Prune can directly derive a compact class-specific edge ViT from the constrained-trained base ViT. Specifically, the pruned FFN module contains two

new weights, $W_1^{(l)'} \in \mathbb{R}^{K^{(l)} \times d}$ and $W_2^{(l)'} \in \mathbb{R}^{d \times K^{(l)}}$, which satisfy:

$$W_1^{(l)'}[k] = \hat{n}_k^{(l)}, \qquad W_2^{(l)'}[:, k] = \sum_{i=1}^{|C_k^{(l)}|} n_{k,2}^{(l,i)}, \qquad k \in [K^{(l)}], \tag{14}$$

where $n_{k,2}^{(l,i)}$ denotes the neuron in $W_2^{(l)}$ corresponding to $n_k^{(l,i)}$. This process is essentially the inverse of network expansion (Ding et al., 2023; Wang et al., 2024b) and adheres to the function-preserving principle (Chen et al., 2016). For the pruned MHA module, its four new weights, $W_{Q|K}^{(l,h)'} \in \mathbb{R}^{q_l' \times d}$, $W_V^{(l,h)'} \in \mathbb{R}^{v_l' \times d}$, and $W_O^{(l,h)'} \in \mathbb{R}^{d \times v_l'}$, can be directly obtained through SVD:

$$W_Q^{(l,h)'} = \left(U_{QK}^{(l,h)}[:, : q_l'] \Sigma_{QK}^{(l,h)}[: q_l', : q_l']\right)^\top \times \sqrt{\frac{q_l'}{q_l}}, \qquad W_K^{(l,h)'} = \left(V_{QK}^{(l,h)}[:, : q_l']\right)^\top, \tag{15}$$

$$W_V^{(l,h)'} = \left(U_{VO}^{(l,h)}[:, : v_l'] \Sigma_{VO}^{(l,h)}[: v_l', : v_l']\right)^\top, \qquad W_O^{(l,h)'} = V_{VO}^{(l,h)}[:, : v_l'], \tag{16}$$

where $U_{QK}^{(l,h)} \Sigma_{QK}^{(l,h)} V_{QK}^{(l,h)\top} = W_{QK}^{(l,h)}$ and $U_{VO}^{(l,h)} \Sigma_{VO}^{(l,h)} V_{VO}^{(l,h)\top} = W_{VO}^{(l,h)}$.

AC-Prune promotes redundancy in FFNs and low-rank structure in MHA before removing parameters, helping preserve useful knowledge during compression while reducing model size.

4 EXPERIMENTS AND ANALYSIS

Models and Datasets. We evaluate AC-Prune on two widely adopted ViT families: DeiT-Base/Small/Tiny (Touvron et al., 2021) and Fast/Mask R-CNN (Swin-T) (Liu et al., 2021). For recognition tasks, we use ImageNet-1K (Russakovsky et al., 2015) and CIFAR-100/10 (Krizhevsky, 2009); for detection and segmentation tasks, we use COCO (Lin et al., 2014). Sub-tasks of varying scales are constructed by randomly sampling classes from each dataset.

Implementations. GFLOPs is used as the metric to compute pruning rates. We use AdamW for constrained training with a batch size of 256 and a learning rate of 10^{-4}.To ensure the models can perform accelerated inference across various edge devices, the pruned dimensions are aligned to multiples of 8 (NVIDIA, 2023). Further details are provided in Appendix A.

Baselines. We compare AC-Prune with five class-agnostic structured pruning baselines: Random Pruning (Gadhikar et al., 2023), NViT (Yang et al., 2023), X-Pruner (Yu & Xiang, 2023), DC-ViT (Zhang et al., 2024), and MDP (Sun et al., 2025), as well as two class-specific methods, Vulcan (Wei et al., 2026a) and NuWa (Wei et al., 2026b). Further details are provided in Appendix B.

4.1 RESULTS

Class-Agnostic vs. Class-Specific. We derive class-specific models from DeiT-Base and calibrate all baselines on the sub-task dataset D_S for a fair comparison. As shown in Table 1, AC-Prune outperforms class-agnostic pruning methods across sub-tasks and pruning rates, with accuracy gains of up to 15.72% and 14.96% over the base ViT and these methods, respectively. At pruning rates of 0.60 and 0.80, AC-Prune exceeds X-Pruner in average accuracy by 3.77% and 10.68%, respectively, showing a greater advantage at higher pruning rates. It also retains 98.54% and 94.97% of the average accuracy of the fine-tuned base ViT on the target sub-tasks, respectively.

Comparison with Existing Class-Specific Methods. As shown in Table 1, AC-Prune achieves the highest group-average accuracy across both pruning rates and all three sub-task sizes. At a pruning rate of 0.60, its accuracy gains over Vulcan and NuWa reach 1.37% and 0.76%, respectively. At a pruning rate of 0.80, these maximum gains increase to 2.20% and 2.16%, demonstrating more pronounced advantages under greater compression while preserving class-specific knowledge.

Generality Across Models and Datasets. We further evaluate AC-Prune on DeiT-Small/Tiny with CIFAR-100/10 and on Fast/Mask R-CNN with Swin-T backbones for COCO detection and segmentation. As shown in Fig. 7, AC-Prune outperforms the base models on recognition tasks at pruning rates below 0.60, and even at 0.90 on some small-scale sub-tasks. On COCO, AC-Prune surpasses

Table 1: Overall performance of AC-Prune and baselines on DeiT-Base across ImageNet-1K sub-tasks of different sizes (25, 50, and 100 classes) at pruning rates of 0.60 and 0.80. The best top-1 accuracy (%) for each size is highlighted in bold, while the second best is underlined. '(FT)' denotes fine-tuning on class-specific data. Results for other pruning rates are provided in Appendix C.

Method	T1/25	T2/25	T3/25	**Avg**	T4/50	T5/50	T6/50	**Avg**	T7/100	T8/100	T9/100	**Avg**
DeiT-Base	79.92	82.56	80.67	81.05	80.48	79.64	84.96	81.69	80.49	84.12	78.58	81.06
DeiT-Base (FT)	96.40	96.96	96.96	96.77	95.00	93.36	94.68	94.35	91.56	93.29	92.84	92.56
Pruning Rate = 0.60												
Random	91.76	92.72	89.69	91.39	86.16	85.60	87.56	86.44	83.11	86.10	80.42	83.21
NViT	83.91	87.12	85.62	85.55	81.64	81.20	80.72	81.19	78.31	80.31	78.12	78.91
X-Pruner	91.68	93.04	93.21	92.64	90.24	88.28	90.16	89.56	84.51	87.12	86.46	86.03
DC-ViT	71.60	83.28	76.92	77.27	69.48	67.20	66.96	67.88	60.06	64.98	65.18	63.41
MDP	91.04	94.80	92.09	92.64	90.32	85.00	91.16	88.83	82.61	84.33	82.86	83.27
Vulcan	95.04	96.24	95.53	95.60	92.16	91.72	92.44	92.11	88.03	90.70	89.02	89.25
Nuwa	95.36	96.88	95.60	<u>95.95</u>	92.32	92.12	93.44	<u>92.63</u>	89.14	91.04	89.42	<u>89.87</u>
AC-Prune	95.64	96.72	96.08	**96.15**	93.04	92.40	93.48	**92.97**	89.40	91.68	90.18	**90.42**
Pruning Rate = 0.80												
Random	71.36	78.48	82.90	77.58	71.92	71.16	73.28	72.12	65.28	67.71	60.98	64.66
NViT	64.16	58.00	60.38	60.85	52.76	53.44	49.04	51.75	41.56	47.62	48.74	45.97
X-Pruner	85.60	86.00	85.62	85.74	76.24	74.00	79.72	76.65	72.62	76.74	75.52	74.96
DC-ViT	54.72	66.24	64.46	61.81	47.52	53.40	50.84	50.59	35.66	35.12	34.66	35.15
MDP	83.68	88.08	84.66	85.47	74.04	74.20	79.88	76.04	61.72	67.81	67.54	65.69
Vulcan	92.24	94.32	92.57	<u>93.04</u>	88.36	88.12	88.52	88.33	81.82	85.20	82.84	83.29
Nuwa	91.60	93.44	91.68	92.24	88.64	87.72	89.08	<u>88.48</u>	81.70	85.56	82.88	<u>83.38</u>
AC-Prune	93.20	95.36	93.84	**94.13**	90.44	89.16	90.72	**90.11**	83.76	87.16	84.58	**85.17**

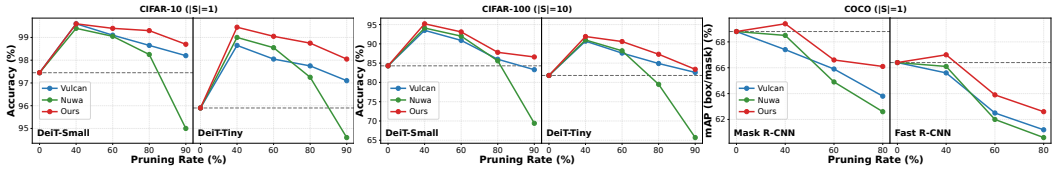


Figure 7: Performance of edge ViTs derived by AC-Prune across different base models and datasets.

the base models at pruning rates of 0.20 and 0.40 and consistently outperforms Vulcan and NuWa across all evaluated pruning rates, demonstrating its broad applicability.

Computational Efficiency. We evaluate AC-Prune-derived models on Jetson Orin NX (batch size 1) and NVIDIA RTX 4090 (batch size 256), averaging results over the nine sub-tasks in Table 1. As shown in Table 2, at pruning rates of 0.20–0.80, these models achieve speedups of 1.27×–5.37× and 1.17×–2.67× over DeiT-Base on the two devices, while reducing FLOPs by 22.88%–81.45%.

Table 2: Comparison of computational efficiency between DeiT-Base and edge ViTs derived by AC-Prune under different R, evaluated on Jetson Orin NX (bz=1) and NVIDIA RTX 4090 (bz=256).

Methods	**Latency (ms)**		**Throughput (image/s)**		**Memory (GiB)**		**#Param**	**#FLOPs**
	Orin NX	RTX 4090	Orin NX	RTX 4090	Orin NX	RTX 4090	(M)	(G)
DeiT-Base	106.44	272.35	9.36	932.20	0.34	2.21	86.57	17.57
AC-Prune (0.20)	84.03 (1.27×)	228.10 (1.17×)	11.88	1113.01	0.27	2.14	67.21	13.55 (↓22.88%)
AC-Prune (0.40)	63.76 (1.67×)	187.58 (1.45×)	15.67	1358.41	0.21	1.82	50.99	10.24 (↓41.72%)
AC-Prune (0.60)	32.24 (3.30×)	146.78 (1.86×)	30.96	1758.76	0.15	1.66	34.09	6.77 (↓61.47%)
AC-Prune (0.80)	19.82 (5.37×)	101.98 (2.67×)	50.45	2495.44	0.08	1.40	16.96	3.26 (↓81.45%)

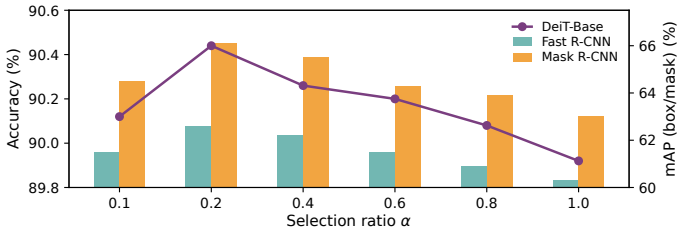


Figure 8: Sensitivity to the token selection ratio α at a pruning rate of 0.80, measured by top-1 classification accuracy on sub-task T4 and box/mask mAP on COCO.

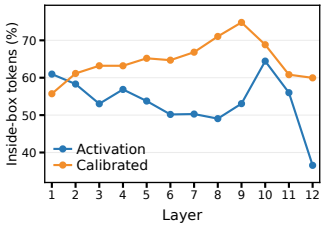


Figure 9: Layer-wise inside-box proportions of selected tokens on sub-task T4.

4.2 Analysis

Effectiveness of Activation Calibration. We evaluate the attention and frequency branches by removing each in turn. As shown in Table 3, removing attention reduces average classification accuracy from 90.11% to 89.15% and box/mask mAP from 62.6/66.1 to 61.4/63.6. Removing frequency reduces these metrics to 89.65% and 60.6/63.2, respectively. Attention contributes more to classification, whereas frequency contributes more to detection and segmentation under the evaluated settings. Combining the cues performs best across three tasks, supporting their complementarity.

Table 3: Ablation of attention (Attn) and frequency (Freq) in activation calibration on sub-tasks T4–T6 and COCO detection and segmentation at a pruning rate of 0.80.

Setting	T4/50	T5/50	T6/50	Avg.	Detection	Segmentation
w/o Attn	89.24	88.36	89.84	89.15	61.4	63.6
w/o Freq	89.96	88.78	90.20	89.65	60.6	63.2
AC-Prune	**90.44**	**89.16**	**90.72**	**90.11**	**62.6**	**66.1**

Table 4: Ablation of ACNC and TNNR on sub-tasks T4–T6 at a target pruning rate of 0.60.

Setting	T4/50	T5/50	T6/50	Avg.
w/o ACNC	7.80	6.16	9.44	7.80
w/o TNNR	82.24	78.76	82.92	81.31
AC-Prune	**93.04**	**92.40**	**93.48**	**92.97**

Sensitivity to the Selection Ratio. At a pruning rate of 0.80, we evaluate the selection ratio α on sub-task T4 and COCO. As shown in Fig. 8, all three tasks achieve their best performance at $\alpha = 0.2$. Classification accuracy varies by only 0.52% over the tested range, whereas dense prediction performance varies more with α. These results indicate that dense prediction tasks are more sensitive to the selection ratio under the evaluated settings. We therefore use $\alpha = 0.2$ in the main experiments.

Interpretability of Activation Calibration. We compare layer-wise inside-box proportions of the top 20% of patch tokens selected by activation magnitude and calibrated activation. On sub-task T4 (Fig. 9), selection based on calibrated activation achieves higher proportions in most layers, indicating greater emphasis on target-region tokens that are relevant to the sub-task. Further interpretability results across additional sub-tasks and layers are provided in Appendix D.

Ablation Study. On sub-tasks T4–T6 at a pruning rate of 0.60 (Table 4), removing ACNC and TNNR reduces average accuracy by 85.17% and 11.67%, respectively. The larger drop without ACNC highlights the importance of preserving FFN knowledge, as FFNs account for roughly two-thirds of ViT parameters. Both components contribute to effective compression.

5 Conclusion

This paper presented AC-Prune, a class-specific structured pruning framework for deploying ViTs on resource-constrained edge devices. We analyzed the limitation of evaluating neuron importance solely through activation magnitudes: background tokens can also produce strong responses, interfering with the selection of category-critical neurons. To address this limitation, AC-Prune calibrates token activation contributions using frequency stability and attention weights to reduce background interference, and aggregates knowledge before pruning through ACNC and TNNR to mitigate irreversible knowledge loss. Experiments with five base ViTs across three visual tasks and four datasets demonstrate that AC-Prune substantially compresses models while achieving superior class-specific performance over the base models and existing pruning methods.

AI USAGE STATEMENT

We used AI tools to improve the language, readability, and clarity of the manuscript. The authors are responsible for reviewing all AI-assisted text and take full responsibility for the final content.

REPRODUCIBILITY STATEMENT

Section 3 describes the proposed method, and Section 4 presents the experimental setup. Appendix A provides implementation details and training hyperparameters, while Appendix B describes the baselines and comparison settings. We will publicly release the code upon paper acceptance.

REFERENCES

Jiajun Cao, Yuan Zhang, Tao Huang, Ming Lu, Qizhe Zhang, Ruichuan An, Ningning Ma, and Shanghang Zhang. MoVE-KD: Knowledge distillation for VLMs with mixture of visual encoders. In *CVPR*, 2025.

Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. In *NeurIPS*, 2021.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2Net: Accelerating learning via knowledge transfer. In *ICLR*, 2016.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *TPAMI*, 2024.

Dahun Choi and Hyun Kim. GradQ-ViT: Robust and efficient gradient quantization for vision transformers. In *AAAI*, 2025.

Ning Ding, Yehui Tang, Kai Han, Chao Xu, and Yunhe Wang. Network expansion for practical training acceleration. In *CVPR*, 2023.

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

Joshua Fixelle. Hypergraph vision transformers: Images are more than nodes, more than edges. In *CVPR*, 2025.

Advait Harshal Gadhikar, Sohom Mukherjee, and Rebekka Burkholz. Why random pruning is all we need to start sparse. In *ICML*, 2023.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2020.

Manas Gupta, Efe Camci, Vishandi Rudy Keneta, Abhishek Vaidyanathan, Ritwik Kanodia, Ashish James, Chuan-Sheng Foo, Min Wu, and Jie Lin. Is complexity required for neural network pruning? A case study on global magnitude pruning. In *CAI*, 2024.

Dongchen Han, Xuran Pan, Yizeng Han, Shiji Song, and Gao Huang. FLatten Transformer: Vision transformer using focused linear attention. In *ICCV*, 2023.

Jian Han, Jinlai Liu, Yi Jiang, Bin Yan, Yuqi Zhang, Zehuan Yuan, Bingyue Peng, and Xiaobing Liu. Infinity∞: Scaling bitwise autoregressive modeling for high-resolution image synthesis. In *CVPR*, 2025.

Changyi He, Yifu Ding, Jinyang Guo, Ruihao Gong, Haotong Qin, and Xianglong Liu. DA-KD: Difficulty-aware knowledge distillation for efficient large language models. In *ICML*, 2025.

Jitesh Jain, Jiachen Li, Mang Tik Chiu, Ali Hassani, Nikita Orlov, and Humphrey Shi. OneFormer: One transformer to rule universal image segmentation. In *CVPR*, 2023.

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

Zhu Liao, Victor Quétu, Van-Tam Nguyen, and Enzo Tartaglione. Can unstructured pruning reduce the depth in deep neural networks? In *ICCV Workshops*, 2023.

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 *ECCV*, 2014.

Xinyu Liu, Houwen Peng, Ningxin Zheng, Yuqing Yang, Han Hu, and Yixuan Yuan. EfficientViT: Memory efficient vision transformer with cascaded group attention. In *CVPR*, 2023.

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 *ICCV*, 2021.

Zhiying Lu, Chuanbin Liu, Xiaojun Chang, Yongdong Zhang, and Hongtao Xie. DHVT: Dynamic hybrid vision transformer for small dataset recognition. *TPAMI*, 2025.

NVIDIA. Train with mixed precision, 2023. URL https://docs.nvidia.com/deeplearning/performance/mixed-precision-training/index.html.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. SAM 2: Segment anything in images and videos. In *ICLR*, 2025.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *IJCV*, 2015.

Shaibal Saha and Lanyu Xu. Vision transformers on the edge: A comprehensive survey of model compression and acceleration strategies. *Neurocomputing*, 2025.

Xuan Shen, Zhao Song, Yufa Zhou, Bo Chen, Jing Liu, Ruiyi Zhang, Ryan A Rossi, Hao Tan, Tong Yu, Xiang Chen, et al. Numerical pruning for efficient autoregressive models. In *AAAI*, 2025.

Cheng Shi, Yizhou Yu, and Sibei Yang. Vision transformers need more than registers. In *CVPR*, 2026.

Xinglong Sun, Barath Lakshmanan, Maying Shen, Shiyi Lan, Jingde Chen, and Jose M Alvarez. MDP: Multidimensional vision model pruning with latency constraint. In *CVPR*, 2025.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021.

Shikhar Tuli and Niraj K Jha. EdgeTran: Device-aware co-search of transformers for efficient inference on mobile edge platforms. *TMC*, 2024.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. DeepNet: Scaling transformers to 1,000 layers. *TPAMI*, 2024a.

Xiao Wang, Yu Jin, Wentao Wu, Wei Zhang, Lin Zhu, Bo Jiang, and Yonghong Tian. Object detection using event camera: A MoE heat conduction based detector and a new benchmark dataset. In *CVPR*, 2025a.

Yite Wang, Jiahao Su, Hanlin Lu, Cong Xie, Tianyi Liu, Jianbo Yuan, Haibin Lin, Ruoyu Sun, and Hongxia Yang. LEMON: Lossless model expansion. In *ICLR*, 2024b.

Zhaoqing Wang, Xiaobo Xia, Runnan Chen, Dongdong Yu, Changhu Wang, Mingming Gong, and Tongliang Liu. LaVin-DiT: Large vision diffusion transformer. In *CVPR*, 2025b.

Ziteng Wei, Qiang He, Feifei Chen, Ranjie Duan, Xiaodan Li, Bin Li, YueFeng Chen, Hui Xue, Hai Jin, and Yun Yang. Vulcan: Crafting compact class-specific vision transformers for edge intelligence. In *ICLR*, 2026a.

Ziteng Wei, Qiang He, Bing Li, Feifei Chen, Hai Jin, and Yun Yang. NuWa: Deriving lightweight class-specific vision transformers for edge devices. In *CVPR*, 2026b.

Chenhao Xu, Chang-Tsun Li, Chee Peng Lim, and Douglas Creighton. HSViT: Horizontally scalable vision transformer. In *IJCNN*, 2025.

Huanrui Yang, Hongxu Yin, Maying Shen, Pavlo Molchanov, Hai Li, and Jan Kautz. Global vision transformer pruning with Hessian-aware saliency. In *CVPR*, 2023.

Yifei Yang, Zouying Cao, and Hai Zhao. LaCo: Large language model pruning via layer collapse. In *Findings of EMNLP*, 2024.

Shengyuan Ye, Jiangsu Du, Liekang Zeng, Wenzhong Ou, Xiaowen Chu, Yutong Lu, and Xu Chen. Galaxy: A resource-efficient collaborative edge AI system for in-situ transformer inference. In *INFOCOM*, 2024.

Hongtian Yu, Yunjie Tian, Qixiang Ye, and Yunfan Liu. Spatial transform decoupling for oriented object detection. In *AAAI*, 2024.

Lu Yu and Wei Xiang. X-Pruner: Explainable pruning for vision transformers. In *CVPR*, 2023.

Hanxiao Zhang, Yifan Zhou, and Guo-Hua Wang. Dense vision transformer compression with few samples. In *CVPR*, 2024.

Jinnian Zhang, Houwen Peng, Kan Wu, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. MiniViT: Compressing vision transformers with weight multiplexing. In *CVPR*, 2022.

Yunshan Zhong, You Huang, Jiawei Hu, Yuxin Zhang, and Rongrong Ji. Towards accurate post-training quantization of vision transformers via error reduction. *TPAMI*, 2025.

A Implementation Details

Table 5 summarizes the hyperparameters used for constrained training. Before determining the retained FFN dimensions $K^{(l)}$, we apply z-score normalization to activation-calibrated neuron importance scores across blocks. Anchor neurons are updated after each batch.

Table 5: Hyperparameters for the constrained training of AC-Prune. Here, R denotes the target pruning rate, and λ denotes the Lagrange multipliers.

Hyperparameter	**Value**
Steps	$6250R^2 + 1250R, \quad R \in (0, 1)$
Optimizer	AdamW
Batch Size	256 (Recognition), 8 (Detection/Segmentation)
Learning Rate (LR)	10^{-4} (Recognition), 5×10^{-5} (Detection/Segmentation)
LR Scheduler	Constant
Weight Decay	0.05
Penalty Parameter ρ	1.0
Multiplier Update Rule	$\lambda_{t+1} = \lambda_t + \rho\, \partial\mathcal{L}/\partial\lambda_t$
Token Selection Ratio α	0.2
Seed	42

B Baseline Methods and Comparison Settings

We describe the five class-agnostic and two class-specific pruning baselines used in our comparisons. Here, e denotes the FFN intermediate dimension; q and v denote the query–key and value–output dimensions of MHA; H denotes the number of attention heads; and d denotes the embedding dimension.

- **Random.** Random pruning operates on the same dimensions as Vulcan: the FFN intermediate dimension e and the MHA query–key and value–output dimensions (q, v). Given a pruning rate, it uniformly prunes these dimensions by randomly selecting neurons to remove, without considering the specific sub-task. We use this class-agnostic method as a lower-bound benchmark for evaluating class-specific model derivation.
- **NViT (Yang et al., 2023).** NViT jointly prunes the FFN intermediate dimension e, the MHA query–key and value–output dimensions (q, v), the number of attention heads H, and the embedding dimension d. It groups weights along each dimension and evaluates their importance using a Hessian-based score. Groups with the lowest scores are iteratively removed until the resource constraints of the target device are satisfied.
- **X-Pruner (Yu & Xiang, 2023).** X-Pruner prunes the FFN intermediate dimension e and the number of attention heads H in MHA. It introduces learnable masks into the ViT and uses sparsity regularization to gradually drive parts of the mask values to zero, identifying structures to remove.
- **DC-ViT (Zhang et al., 2024).** DC-ViT aims to achieve the target pruning rate while modifying as few blocks as possible. It evaluates block importance and, within the selected blocks, removes the entire MHA module and randomly prunes the FFN intermediate dimension e at a uniform rate. Among the baselines considered here, DC-ViT uniquely includes whole-module removal.
- **MDP (Sun et al., 2025).** Similar to NViT, MDP jointly prunes the embedding dimension d, the number of attention heads H, the MHA query–key and value–output dimensions (q, v), and the FFN intermediate dimension e. It formulates pruning under latency budgets as a mixed-integer nonlinear programming (MINLP) problem, using Hessian-based importance scores and a precomputed latency lookup table (LUT) to determine the pruning configuration.
- **Vulcan (Wei et al., 2026a).** Vulcan derives class-specific ViTs through pruning-oriented post-training. Its class-centric neuron collapse (CCNC) groups FFN neurons by their weights and uses strongly activated neurons on the target classes as anchors. Regularization encourages neurons within each group to converge toward their anchor, allowing their contributions to be aggregated before redundant neurons are removed. In parallel, truncated nuclear norm regularization (TNNR) promotes low-rank MHA weight products, which are subsequently compressed using singular value decomposition (SVD). Thus, Vulcan prepares the FFN and MHA structures for compression through a train-then-prune procedure.
- **NuWa (Wei et al., 2026b).** NuWa combines self-knowledge purification (SKP) with optimization-based fast pruning (OFP). SKP learns binary masks to remove weights that harm target-class performance, producing an anchor model. OFP then compresses MHA through SVD-based low-rank approximation and prunes FFN neurons according to their mean activations on class-specific data. Rather than simply deleting the discarded neurons' contributions, it reconstructs the FFN output weights through a closed-form optimization using calibration features. This enables model derivation without post-pruning retraining in the original method.

Comparison Settings. For class-agnostic baselines, we retrain the pruned models until convergence. For class-specific methods, Vulcan and NuWa, we reproduce their methods strictly following the descriptions and experimental settings in the original papers.

C Results at Other Pruning Rates

We further compare AC-Prune with baseline methods at pruning rates of 0.20 and 0.40 to evaluate its performance under moderate pruning conditions. NuWa is omitted at a pruning rate of 0.20 because its SKP stage learns task-dependent pruning rates and does not guarantee this fixed target across sub-tasks. As shown in Table 6, AC-Prune achieves the best average accuracy across different sub-task sizes. Specifically, at a pruning rate of 0.20, AC-Prune improves accuracy over the base ViT by 13.71% on average, while its group-average accuracy gains over the best-performing class-agnostic and class-specific baselines reach 0.73% and 0.66%, respectively. At a pruning rate of 0.40, the average gain over the base ViT is 12.99%, and the maximum group-average gains over the two baseline categories reach 1.99% and 0.39%, respectively. These results further demonstrate the effectiveness of AC-Prune under moderate pruning conditions.

Table 6: Overall performance of AC-Prune and baselines on DeiT-Base across ImageNet-1K subtasks of different sizes (25, 50, and 100 classes) at pruning rates of 0.20 and 0.40. The best top-1 accuracy (%) for each size is highlighted in bold, while the second best is underlined. '(FT)' denotes fine-tuning on class-specific data.

Method	T1/25	T2/25	T3/25	**Avg**	T4/50	T5/50	T6/50	**Avg**	T7/100	T8/100	T9/100	**Avg**
DeiT-Base	79.92	82.56	80.67	81.05	80.48	79.64	84.96	81.69	80.49	84.12	78.58	81.06
DeiT-Base (FT)	96.40	96.96	96.96	96.77	95.00	93.36	94.68	94.35	91.56	93.29	92.84	92.56
Pruning Rate = 0.20												
Random	95.60	96.48	96.40	96.16	93.84	93.36	94.60	93.93	91.41	93.07	91.44	91.97
NViT	94.96	96.32	96.01	95.76	92.96	92.72	94.32	93.10	91.27	92.83	91.84	91.98
X-Pruner	96.56	96.48	96.88	96.62	94.40	93.00	95.40	94.26	91.16	92.65	91.70	91.84
DC-ViT	95.52	95.76	94.57	95.28	91.40	93.16	94.00	92.85	84.73	88.34	90.84	87.97
MDP	95.44	96.56	94.73	95.58	92.60	88.00	92.48	91.03	84.50	92.79	90.74	89.34
Vulcan	96.48	97.12	97.04	96.88	95.00	93.24	94.60	94.28	91.49	92.91	91.74	92.05
AC-Prune	96.72	97.84	97.36	**97.31**	94.96	94.64	95.16	**94.92**	92.02	93.58	92.52	**92.71**
Pruning Rate = 0.40												
Random	94.64	95.68	94.48	94.93	91.84	90.84	92.48	91.72	87.94	90.41	89.12	89.16
NViT	92.08	94.24	94.17	93.50	90.60	90.00	91.52	90.71	87.23	90.10	87.88	88.40
X-Pruner	94.76	94.68	95.01	94.82	92.34	91.60	93.28	92.41	88.95	91.67	89.04	89.89
DC-ViT	86.80	89.12	88.66	88.19	83.84	80.60	87.52	83.99	80.91	83.73	83.16	82.60
MDP	95.36	96.40	93.69	95.15	93.40	89.76	90.60	91.25	88.89	88.12	89.36	88.79
Vulcan	94.80	95.68	94.97	95.15	92.76	91.52	93.60	92.63	89.71	91.59	89.84	90.38
NuWa	96.16	97.04	96.72	**96.64**	93.96	93.52	94.80	94.09	90.72	92.38	91.36	91.49
AC-Prune	96.16	97.28	96.48	**96.64**	94.12	93.60	95.05	**94.26**	91.08	92.94	91.62	**91.88**

D Further Analysis of Activation Calibration

To further examine the token selection behavior after activation calibration across sub-tasks, we compare its layer-wise inside-box proportions with those of activation-based selection (Fig. 10).

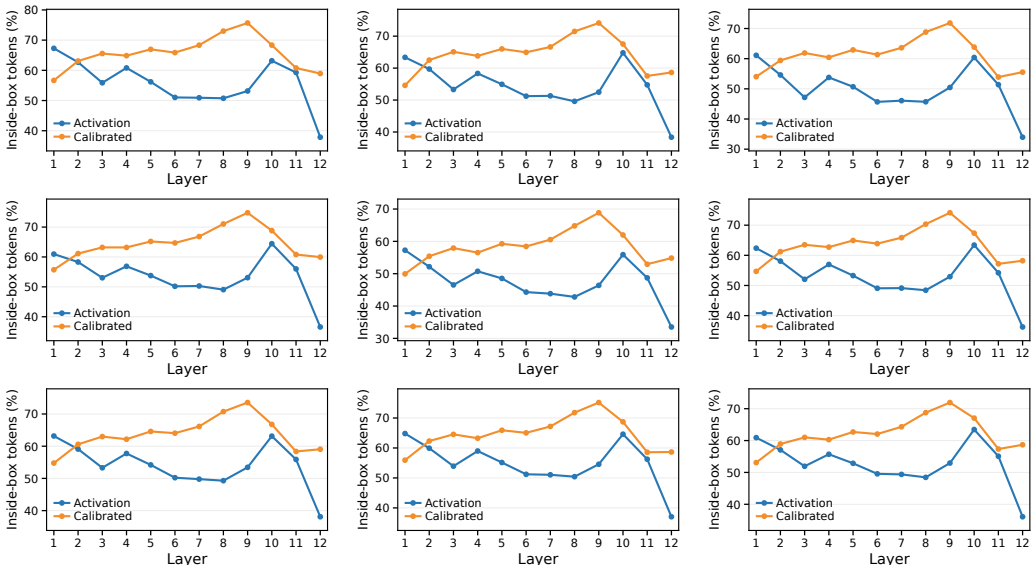


Figure 10: Layer-wise inside-box proportions of selected tokens across sub-tasks T1–T9. Blue denotes activation-based selection, while orange denotes calibrated selection.