AWP: Activation-based Window Pruning for Gigapixel Object Detection

Xiang Li[1,2,3*] **Wenxi Li**[1*] **Yuetong Wang**[4*]
Chen Zhang[7] **Chenyang Lyu**[6] **Haozhe Lin**[2]
Fan Zhang[5] **Guiguang Ding**[2,5†] **Yuchen Guo**[2†]

[1]KLATASDS-MOE, School of Statistics,
East China Normal University, Shanghai, China
[2]Beijing National Research Center for Information Science and Technology,
Tsinghua University, Beijing, China
[3]Department of Automation, Tsinghua University, Beijing, China
[4]Xiuzhong College, Tsinghua University, Beijing, China
[5]School of Software, Tsinghua University, Beijing, China
[6]Alibaba Token Hub, Alibaba Group, Hangzhou, China
[7]Hangzhou Dianzi University

{xiang-li22,wangyuet23,f-zhang22}@mails.tsinghua.edu.cn
{linhz,dinggg}@tsinghua.edu.cn chenzhang@hdu.edu.cn
wxli@sfs.ecnu.edu.cn lyuchenyang.dcu@gmail.com
yuchen.w.guo@gmail.com

Abstract

Gigapixel object detection in High-Resolution Wide (HRW) images faces extreme spatial sparsity where targets often occupy less than 5% of the image area while computation is wasted on vast uninformative regions. Existing token selection methods require either complex evolutionary search or training additional learnable modules. We present AWP (Activation-based Window Pruning), built on the discovery that feed-forward network (FFN) activations in trained vision transformers naturally encode window importance, enabling a parameter-free, training-free selection mechanism without additional modules. Leveraging this intrinsic signal, AWP performs window pruning with consistent improvements. On PANDA, fine-tuned AWP at k=0.5 improves over SparseFormer at k=0.7 by **1.6 AP_{50} points** and 4.3 small-object AP points while reducing backbone FLOPs by **26.2%**. When applied to Swin Transformer, AWP demonstrates strong generalization, retaining 95.7% of baseline AP_{50} with 50% windows pruned without retraining, and the same signal also transfers to other backbones (PVTv2, RMT) and to additional detection benchmarks. Remarkably, AWP enables zero-shot application by directly replacing learned selection modules, improving baseline AP_{50} by 0.6 points without retraining. Our work reveals that window-based vision transformers inherently encode spatial importance through FFN activations, offering a simple yet powerful alternative to existing selection mechanisms, especially for High-Resolution Wide images.

1 Introduction

Object detection, a fundamental task in computer vision, aims to identify and localize objects of interest within an image. Over the past decade, the field has witnessed tremendous progress, with

*These authors contributed equally.
†Corresponding authors: Guiguang Ding (dinggg@tsinghua.edu.cn) and Yuchen Guo (yuchen.w.guo@gmail.com).

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

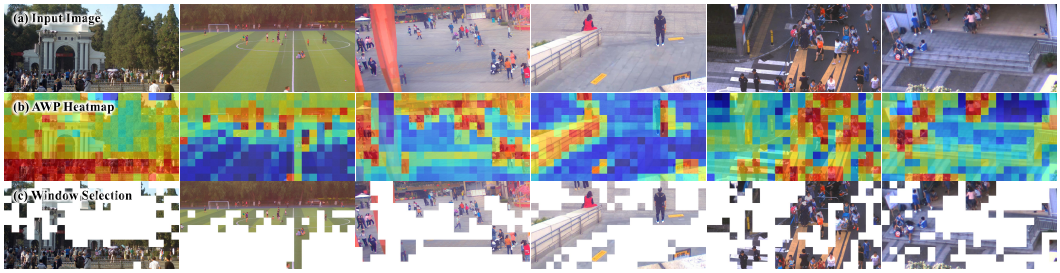

Figure 1: **AWP aims at the right targets for gigapixel detection.** Our method identifies important windows through FFN activation patterns. The heatmap (middle row) reveals that AWP naturally distinguishes object-containing regions (red, high activation) from empty backgrounds (blue, low activation). Based on these intrinsic signals, AWP selects high-importance windows for processing (bottom row, visible regions) while pruning uninformative areas (white), achieving efficient detection with negligible accuracy loss. Zero-shot visualization on Swin Transformer [1] (keep ratio=0.5 for illustration) demonstrates AWP's effectiveness across diverse scenes.

methods developed on benchmarks like MS COCO [2] achieving remarkable performance. Iconic architectures have evolved from R-CNN variants [3–5] to widely-adopted YOLO series [6–10], and Transformer-based models [11, 12]. These advances mainly concern "close-up" images with a limited field of view, where objects occupy much of the frame.

Recent advances in imaging systems for aerial surveillance, autonomous driving, and smart city management have propelled a new frontier: object detection in High-Resolution Wide (HRW) shots [13–17], where a single gigapixel frame can encapsulate vast scenes of several square kilometers into billions of pixels. This shift from object-centric close-ups to scene-level vistas introduces **extreme information sparsity**: with objects of interest often occupying less than 5% of the total area, detectors must extract critical features from vast expanses of background, leading to high rates of false positives in empty regions and false negatives where small or low-contrast objects are missed amid the surrounding noise. This motivates *sparse processing*: allocating computation to informative regions instead of processing the scene uniformly.

Recent work has explored token or window selection mechanisms to address this challenge. SparseFormer [15], the current state-of-the-art on PANDA [13], employs a learned ScoreNet module that computes window importance based on appearance variance. Variance-based scoring can allocate computation to high-contrast textures outside annotated target regions. AWP instead ranks the spatial response learned for the detection task. Other approaches like SparseViT [18] employ evolutionary search on datasets like COCO [2] and nuScenes [19], though such methods require substantial engineering effort and lack validation on true HRW scenarios where spatial sparsity is far more extreme. This raises the question: *can we find a simpler solution for window selection that works across architectures and requires neither learned modules nor complex search?*

In this paper, we make a surprising discovery: **the activation of Feed-Forward Network (FFN) layers naturally encodes window importance in trained models.** Object and background windows develop distinct FFN activation magnitudes during training, without explicit supervision.

Leveraging this intrinsic signal, we propose **AWP (Activation-based Window Pruning)**, a parameter-free window selection mechanism. Remarkably, AWP can directly replace learned selection modules like ScoreNet without retraining, immediately improving AP_{50} by 0.6 points. With fine-tuning, AWP improves SparseFormer AP_{50} by 2.1 points at k=0.7 on PANDA. At k=0.5, it improves over the same baseline by 1.6 points while reducing backbone FLOPs by 26.2%. AWP also generalizes across architectures: when applied to Swin Transformer, it retains 95.7% of baseline AP_{50} with 50% windows pruned (zero-shot).

Our main contributions are:

- We discover that FFN activation mean serves as an intrinsic signal for window importance, eliminating the need for learned scoring modules.

- We propose AWP, a parameter-free window selection method that outperforms learned approaches and enables zero-shot application, with further improvements through fine-tuning.
- We validate AWP on four backbones (SparseFormer, Swin Transformer, PVTv2, RMT) and across both gigapixel (PANDA) and dense (COCO) detection benchmarks, with targeted ablations on pruning ratios, activation functions, and aggregation methods.

2 Related Work

Object Detection in Gigapixel Images. Object detection on close-up datasets like PASCAL VOC [20] and MS COCO [2] has achieved remarkable progress, with architectures evolving from two-stage detectors [3, 5, 21] to one-stage methods [6, 8, 22, 23] and Transformer-based approaches [11, 12]. However, these methods are ill-suited for High-Resolution Wide (HRW) detection where extreme spatial sparsity dominates. The emergence of gigapixel imaging systems has led to HRW-specific benchmarks like PANDA [13] and DOTA [14]. Early patch-based strategies [24–26] struggled with incomplete detections and high costs. Recent advances like SparseFormer [15] and SaccadeDet [27] employ sparse processing, selecting salient regions for intensive computation. The added parameters and training complexity of learned selection motivate our study of intrinsic signals.

Efficient Vision Transformers. Vision Transformers [28, 29] suffer from quadratic complexity, leading to hierarchical window-based architectures like Swin Transformer [1, 30], which achieves linear complexity through non-overlapping window attention. Various efficiency-oriented variants have emerged, including token merging [31, 32], adaptive selection [33–35], and dynamic resolution [36, 37]. DynamicViT [33] and AdaViT [34] predict token importance through lightweight prediction modules, introducing additional parameters and computational overhead. SparseViT [18] demonstrates that window-level pruning achieves measurable speedup using evolutionary search and L2 activation magnitude, but this requires substantial engineering effort and has been validated primarily on COCO [2] and nuScenes [19], not HRW scenarios.

Window and Token Selection Mechanisms. The core challenge lies in determining *which* regions to process. *Learned mechanisms* employ trainable modules: ScoreNet [15] uses variance-based scoring, which can be misled by complex backgrounds (e.g., foliage), wasting computation on uninformative regions. DGE [38] uses attention-based pooling but lacks flexibility for diverse architectures. *Search-based methods* like SparseViT [18] avoid learned modules but require expensive evolutionary search. *Heuristic approaches* use simple criteria like magnitude thresholding but often underperform. FFN activations provide an intrinsic signal that rivals learned approaches without additional training or search. We compare these parameter-free alternatives in Section 4.5.

3 Method

3.1 Overview of AWP

Core Insight. We propose Activation-based Window Pruning (AWP), a simple yet effective method for sparse processing in window-based vision transformers. The core insight is that **FFN activation patterns naturally encode window importance** in trained models—windows containing objects exhibit distinct activation magnitudes compared to background windows. By capturing and aggregating these activations, we can identify important regions without any learned scoring modules.

Three-step Pipeline. AWP operates in three steps: (1) during the forward pass through the transformer blocks, we read out the FFN intermediate activations; (2) for each window, we aggregate the per-token activations (via mean) to obtain a window-level importance score; (3) we retain the top-k windows for subsequent processing and prune the rest.

Zero-shot and Fine-tuned Use. AWP introduces *zero additional parameters* and can be applied *zero-shot* to pretrained models, directly replacing learned selection modules (e.g., ScoreNet in SparseFormer) without any retraining; a short fine-tuning (6–8 epochs) further improves performance as the model adapts to the new sparse regime. We demonstrate AWP's effectiveness on four window-based vision transformers (SparseFormer, Swin Transformer, PVTv2, RMT) and three detection benchmarks (PANDA, COCO, DOTA).

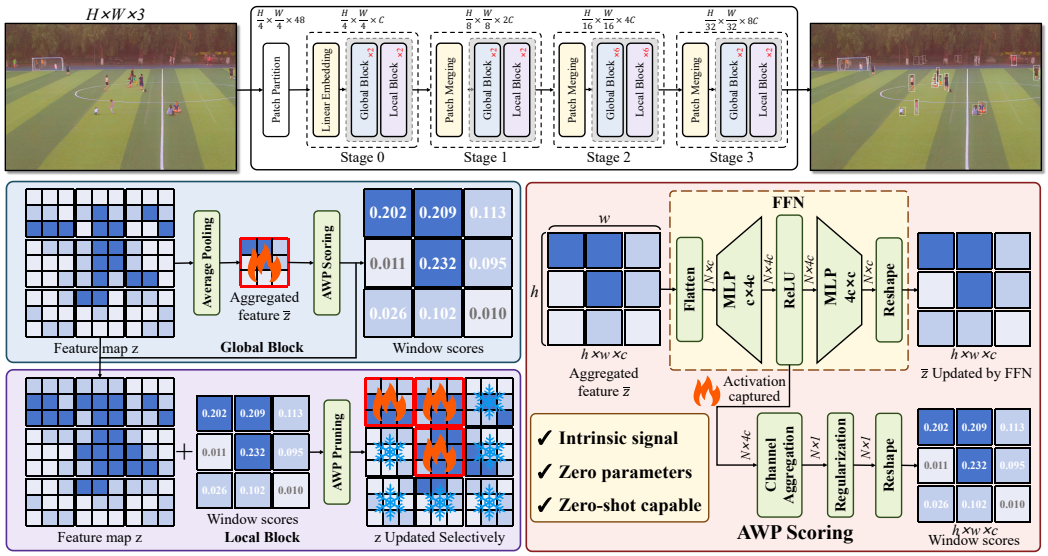

Figure 2: **AWP applied to SparseFormer for gigapixel detection.** Our method captures FFN activations during forward pass to compute window importance scores without additional parameters. AWP replaces the learned ScoreNet in SparseFormer: after global attention aggregates coarse features, AWP scores windows based on FFN activation mean (with regularization), then local attention processes only high-importance windows selected by sparse processing. The orange flame icon indicates where activations are captured. The signal emerges during training and supports parameter-free, zero-shot pruning across architectures.

3.2 AWP for SparseFormer

SparseFormer [15] follows a hierarchical architecture with four stages, each containing Global Attention blocks for coarse-grained features and Local Attention blocks for fine-grained processing on selected windows. AWP replaces its learned ScoreNet within the same sparse processing framework.

Feature Aggregation for Global Attention. Following SparseFormer's design, each stage begins with global attention that operates on aggregated window features. Given input feature map $z \in \mathbb{R}^{H\times W\times C}$, we partition it into non-overlapping windows of size $M \times M$. For a window at (x, y), we average token features independently in each channel (channel indices omitted):

$$\bar{z}_{x',y'} = \frac{1}{M^2} \sum_{\Delta x=0}^{M-1} \sum_{\Delta y=0}^{M-1} z_{x+\Delta x, y+\Delta y}, \tag{1}$$

where $x' = x/M$ and $y' = y/M$ are the coordinates in the aggregated feature map, corresponding to the top-left coordinate (x, y) of a window in the original feature map z. This produces aggregated features $\bar{z} \in \mathbb{R}^{\frac{H}{M}\times\frac{W}{M}\times C}$, reducing tokens by M^2 to enable efficient global attention.

We upsample window-level Multi-Head Self-Attention (MSA) output to the original resolution:

$$z^{\text{global}} = \text{Upsample}(\text{MSA}(\text{LN}(\bar{z}))), \tag{2}$$

These global features are merged with the original feature map via a residual connection:

$$z \leftarrow z + z^{\text{global}}. \tag{3}$$

AWP Scoring via FFN Activations. After global attention, AWP scores windows for local attention directly from FFN activations, replacing ScoreNet's learned variance-based scoring.

Within each global block, we read out the intermediate FFN activations $A \in \mathbb{R}^{H\times W\times C'}$ produced by the first linear projection followed by the nonlinearity, where C' is the FFN hidden dimension. We aggregate activations from the first global block of each stage to score each window:

$$s_{i,j} = \frac{1}{M^2 \cdot C'} \sum_{\Delta x,\Delta y} \sum_{c} |A_{i\cdot M+\Delta x, j\cdot M+\Delta y, c}|, \tag{4}$$

where $s_{i,j}$ is the raw importance score for window at position (i, j).

After channel aggregation, we normalize scores for consistency across images and architectures:

$$\hat{s}_{i,j} = \frac{s_{i,j} - \mu}{\sigma + \epsilon}, \tag{5}$$

Here, μ and σ denote the mean and standard deviation of window scores in the current feature map, while ϵ stabilizes normalization.

Local full-resolution FFNs and pooled global FFNs can require opposite ranking directions. Section 3.4 explains this distinction and the training-free calibration used to select the direction.

Sparse Window Selection and Processing. Given window scores $\{s_{i,j}\}$ and a keep ratio $k \in (0, 1]$, we select the top-N_k windows where $N_k = k \cdot N$ and $N = \frac{H}{M} \times \frac{W}{M}$ is the total number of windows. We construct a binary mask $\mathcal{M} \in \{0, 1\}^N$ indicating selected windows.

For efficient sparse processing, we gather selected window features into a compact representation:

$$\hat{Z}_s = \text{gather}(Z, \mathcal{M}), \tag{6}$$

where $Z \in \mathbb{R}^{N \times D}$ is the window feature matrix ($D = M^2 \cdot C$), and $\hat{Z}_s \in \mathbb{R}^{N_k \times D}$ contains only selected windows.

The sparse features $\hat{Z}_s$ are then processed by the standard local (shifted) window attention blocks [1]. We scatter processed features to their original positions and retain all unselected windows:

$$\hat{Z} = \text{scatter}(\hat{Z}_s, \mathcal{M}) + (1 - \mathcal{M}) \odot Z, \tag{7}$$

Here, $\odot$ is element-wise multiplication; scattering preserves spatial structure.

Zero-shot Replacement of ScoreNet. A key advantage of AWP is that it can directly replace SparseFormer's ScoreNet without retraining. Given a trained SparseFormer model, we simply: (1) Remove the ScoreNet module and its associated loss terms. (2) Read out the FFN intermediate activations. (3) Replace the scoring step with Equation (4). This zero-shot application immediately improves performance (see Section 4), demonstrating that FFN activations provide a more reliable importance signal than learned variance-based scoring.

3.3 AWP for Swin Transformer

To demonstrate AWP's generality, we apply it to the standard Swin Transformer [1] for object detection. Swin Transformer builds hierarchical representations through patch merging and employs shifted window partitioning to enable cross-window connections while maintaining linear complexity. Inspired by SparseViT [18], we modify Swin's implementation to enable window-level pruning.

Swin alternates regular (W-MSA) and shifted (SW-MSA) window attention in each pair of blocks [1].

Window-wise Execution for Pruning. The original Swin applies MHSA at the window level but executes FFN and Layer Normalization (LN) on the entire feature map, requiring reshaping between operations. Following SparseViT, we execute FFN and LN window-wise, so entire windows can be dropped without complex indexing.

AWP Integration with Shifted Windows. We compute window importance using FFN activations from the first block in each stage, score windows via Eq. (4) followed by the normalization in Eq. (5), and select the top-k windows in descending order (the polarity returned by the calibration in Appendix F for Swin Transformer).

The first block (W-MSA) of each stage runs densely to produce the window scores; subsequent blocks (both W-MSA and SW-MSA) process only the selected windows, with the shift applied to the selected partition before SW-MSA computation. Pruned windows reuse their features from the previous layer, preserving spatial structure for cross-window connections.

3.4 Activation Mechanism and Polarity Calibration

Why absolute-mean activation? For an FFN activation a_p, first-order Taylor pruning motivates the saliency $|g_p a_p|$, where $g_p = \partial \mathcal{L}/\partial a_p$ [39]. If gradient magnitudes are approximately constant within a stage, aggregating this proxy over window $\mathcal{W}$ gives

$$T_{\mathcal{W}} = \sum_{p \in \mathcal{W}} |g_p a_p| \approx |g| \sum_{p \in \mathcal{W}} |a_p| = |g| M^2 C' s_{i,j}. \tag{8}$$

Thus the absolute mean provides the same ranking as aggregate magnitude for equal-sized windows, whereas max or L_2 emphasizes peaks. FFN responses are spatially aligned, enabling direct regional aggregation without reducing attention across heads or object queries.

Perturbation evidence and polarity. We masked individual stage-1 output windows and measured relative neck-feature L_2 perturbation on 5 PANDA images (40 random windows each). Across these 200 windows, Spearman correlation with the absolute-mean score is 0.749 for Swin and -0.384 for SparseFormer ($p < 3 \times 10^{-8}$ for both); reversing SparseFormer's ranking gives 0.384. A pooled global branch transplanted into Swin exhibits lower foreground activations while its local branch retains higher foreground activations, supporting a role for input granularity (Appendix E).

Automatic unlabeled calibration. For up to 10 images, we compare neck features from the dense detector with two AWP variants at k=0.5, retaining the highest or lowest scores. Each image votes for the direction with greater cosine similarity to the dense reference; majority vote fixes the inference direction without training or labels. Calibration runs once per detector and target domain. Subsequent images reuse the chosen direction, so the dense reference and the two candidate passes are not repeated during ordinary inference. Every three-image subset of the ten-image calibration sets recovers the selected direction on all four tested backbones (Appendix F).

4 Experiments

4.1 Experimental Setup

Datasets. Our evaluation is conducted on the PANDA [13] benchmark, the first human-centric gigapixel-level dataset for HRW detection. PANDA contains 18 scenes (13 training, 5 testing) and over 15,974.6k bounding boxes, with extreme spatial sparsity and scale variation.

Evaluation Metrics. AP_{50} uses IoU 0.50; COCO AP averages IoU thresholds 0.50:0.05:0.95. PANDA size groups are small ($< 96 \times 96$), medium (96×96 to 288×288), and large ($> 288 \times 288$). AP values use a 0–100 scale; gains are absolute points and retention is relative to the stated baseline. FLOPs use 1280×800 patches. Appendix G reports full-detector latency, throughput, and peak memory.

Implementation Details. We implement our method using MMDetection [40] framework. For SparseFormer experiments, we use the DINO detector [12] as the detection head with SparseFormer backbone. For Swin Transformer experiments, we use the same DINO head with Swin-T backbone. Following SparseFormer's training protocol, all models are trained with multi-scale slicing strategy: images are divided into grids of 16×16, 8×8, 4×4, and 2×2 respectively, with slices containing no objects removed during training. The standard training schedule is 36 epochs from scratch. For fine-tuning, we initialize from pretrained checkpoints and train with AWP for 6–8 additional epochs.

4.2 Main Results on PANDA

We compare AWP against state-of-the-art methods on the PANDA benchmark in Table 1. Our experiments include both zero-shot and fine-tuned variants to demonstrate AWP's flexibility.

Zero-shot Performance. At $k = 0.7$, zero-shot AWP achieves 78.6 AP_{50} and 53.0 AP_S, improving over SparseFormer by 0.6 and 2.2 points, respectively, at similar FLOPs (74.0G vs. 75.7G). This supports FFN activations as a more reliable importance signal than learned variance-based scoring.

Fine-tuned Performance. With 8 epochs of fine-tuning at $k = 0.7$, AWP reaches 80.1 AP_{50} and 57.2 AP_S, gains of 2.1 and 6.4 points over SparseFormer at the same keep ratio. At $k = 0.5$, AWP retains 79.6% AP_{50} while reducing FLOPs by 26.2%, still surpassing the baseline at $k = 0.7$.

Table 1: **Comparison with state-of-the-art methods on PANDA benchmark.** SF denotes SparseFormer. "Keep" indicates the window keep ratio. All AWP methods use DINO [12] detection head. FLOPs use a 1280×800 window. †: SparseFormer [15] results; **bold**: best results.

Method	Epochs	Backbone	Keep	AP_{50}	AP_S	AP_M	AP_L	FLOPs (G)
Prior State-of-the-Art Methods								
Faster RCNN [5]	36	ResNet-101	1.0	–	19.0	55.2	74.4	283.1
ClusDet [41]	36	ResNet-50	1.0	71.8	21.9	69.6	78.2	207.1
PAN [26]	36	ResNet-50	1.0	71.5	25.6	71.9	76.8	207.1
GigaDet [42]	36	CSP-DarkNet-53	1.0	68.4	21.0	59.9	76.2	92.2
DINO [12]	36	Swin-T	1.0	60.6	36.7	61.2	64.9	132.8
DINO+SparseNet†	36	SparseNet	0.7	74.6	38.1	75.4	79.7	107.5
DINO+SparseFormer†	36	SparseFormer	0.7	78.0	50.8	78.1	82.3	75.7
Swin Transformer + AWP (Ours) - Zero-shot								
Swin-T Baseline	36	Swin-T	1.0	75.0	32.0	72.0	82.8	95.8
Swin-T + AWP (Zero-shot)	+0	Swin-T	0.7	73.0	30.4	71.2	80.8	77.8
Swin-T + AWP (Zero-shot)	+0	Swin-T	0.5	71.8	29.8	70.6	79.6	65.3
Swin Transformer + AWP (Ours) - Fine-tuned								
Swin-T + AWP (Fine-tuned)	+8	Swin-T	0.7	73.6	31.0	71.6	81.3	77.8
Swin-T + AWP (Fine-tuned)	+8	Swin-T	0.5	72.5	30.4	71.1	80.2	65.3
SparseFormer + AWP (Ours) - Zero-shot								
SF + AWP (Zero-shot)	+0	SparseFormer	0.7	78.6	53.0	78.4	82.5	74.0
SF + AWP (Zero-shot)	+0	SparseFormer	0.5	77.6	51.2	77.6	81.8	**55.9**
SparseFormer + AWP (Ours) - Fine-tuned								
SF + AWP (Fine-tuned)	+8	SparseFormer	0.7	**80.1**	**57.2**	79.4	**83.6**	74.0
SF + AWP (Fine-tuned)	+8	SparseFormer	0.5	79.6	55.1	**79.7**	83.0	**55.9**

Comparison with State-of-the-Art. Table 1 shows that AWP establishes a new state-of-the-art on PANDA: at $k = 0.5$, our DINO + SparseFormer + AWP reaches 79.6% AP_{50} with only 55.9G FLOPs, jointly improving accuracy and efficiency over prior HRW detectors.

4.3 Generalization to Swin Transformer

To demonstrate AWP's generality beyond SparseFormer, we apply it to standard Swin Transformer on PANDA (Table 1). Zero-shot AWP at $k = 0.7$ retains 97.3% of the dense Swin-T baseline (75.0% $\rightarrow$ 73.0% AP_{50}), and even at $k = 0.5$ it keeps 95.7% of baseline performance. Short fine-tuning (8 epochs) at $k = 0.5$ further recovers performance to 96.7% retention, confirming that AWP transfers to general-purpose backbones beyond HRW-specific architectures.

4.4 Generalization to Other Backbones and Datasets

To probe whether AWP transfers beyond gigapixel scenes and beyond Swin-style backbones, we additionally apply it *zero-shot* to three different vision transformer backbones on COCO [2]: Swin-T, PVTv2-B0 [43], and RMT-S [44]. For each backbone we directly insert AWP into the publicly available pretrained detector without any retraining. COCO AP for dense inference / k=0.7 / k=0.5 is 42.70 / 41.90 / 40.20 for Swin-T, 37.10 / 34.95 / 34.10 for PVTv2-B0, and 49.00 / 46.58 / 43.22 for RMT-S. All three lose less than 2.5 AP points at k=0.7; lower keep ratios allocate less computation to object-containing regions. The full per-keep-ratio AP table and qualitative heatmaps for all three backbones are provided in Appendix C (Table 8 and Figure 6). Beyond COCO, we further visualize AWP zero-shot on DOTA-v1.0 [14], an aerial object-detection benchmark (Appendix D); the same activation pattern emerges despite the substantial domain shift from natural images. Together, these results indicate that the FFN-activation signal exploited by AWP is not specific to one architecture, dataset, or sparsity regime, with the largest computational savings in sparse gigapixel scenes.

Table 2: **Effect of pruning ratio.** ScoreNet vs. AWP on SparseFormer at different keep ratios after 8 epochs fine-tuning. AWP outperforms ScoreNet at every tested keep ratio.

Keep	AP_{50}		AP_S	
	ScoreNet	AWP	ScoreNet	AWP
1.0	80.7	80.7	56.1	56.1
0.9	79.0	80.5	53.4	57.6
0.7	78.0	80.1	50.8	57.2
0.5	77.9	79.6	46.0	55.1
0.3	77.4	78.3	45.8	50.0
0.1	75.6	77.3	42.3	45.7

Table 3: **Activation function ablation.** Evaluated at $k = 0.7$ with mean aggregation after 8 epochs fine-tuning. All activation functions outperform the learned ScoreNet baseline.

Activation	AP_{50}	AP_S	AP_M	AP_L
ReLU	**80.1**	**57.2**	**79.4**	83.6
LeakyReLU	80.0	56.8	79.2	**83.8**
GELU	79.7	55.0	78.8	83.3
SiLU	79.3	53.2	78.1	83.0
ELU	79.1	52.3	78.2	82.7
ScoreNet (baseline)	78.0	50.8	78.1	82.3

4.5 Comparison with Other Parameter-Free Heuristics

To position AWP relative to existing parameter-free importance signals, we compare it against two representative alternatives on the same SparseFormer backbone, evaluated zero-shot on PANDA: the official *ToMe* [31] token merger plugged into the backbone, and a *SparseViT-style* [18] window score that uses the per-window L2 norm of intermediate features. For SparseViT-style and AWP, we share the same window-pruning pipeline and only vary the importance score. Table 4 reports AP_{50} at three keep ratios.

Table 4: **Comparison with parameter-free heuristics on SparseFormer (zero-shot, PANDA, AP_{50}).** ToMe is the official token merger plugged into the backbone; SparseViT-style and AWP share the same window-pruning pipeline and only vary the importance score. AWP is consistently best across all reported keep ratios.

Keep ratio	ToMe [31]	SparseViT-style [18]	AWP (Ours)
0.9	78.2	78.5	**79.4**
0.7	77.8	78.0	**78.6**
0.5	77.1	77.0	**77.6**

AWP outperforms both alternatives at every keep ratio, with the largest gap at k=0.9 (+0.9–1.2 AP_{50}). The two key conceptual differences explain this: (i) AWP operates at the *window* level rather than per token, matching the natural granularity of window-based ViTs; (ii) the importance signal comes from *FFN activation magnitude* rather than token-similarity (ToMe) or feature norms (SparseViT-style). AWP also requires neither evolutionary search (as in SparseViT) nor a similarity-graph computation (as in ToMe); the only configuration is a one-time, training-free polarity check on up to 10 unlabeled images (Appendix F). Appendix H reports variability across PANDA scenes and additional selector comparisons on Swin-T.

4.6 Ablation Studies

Unless stated otherwise, we ablate AWP's design choices on SparseFormer with keep ratio $k = 0.7$.

Effect of Pruning Ratio. Table 2 shows AWP's performance across different keep ratios. As expected, accuracy decreases with more aggressive pruning, but AWP maintains graceful degradation. Interestingly, moderate pruning ($k = 0.9$) achieves the highest small object AP (57.6%), outperforming no pruning, suggesting that removing a small fraction of background windows helps the model focus on object regions. At $k = 0.5$, we still achieve 79.6% AP_{50}, demonstrating that half the windows can be safely pruned with minimal accuracy loss.

Activation Function Choice. We compare five activation functions in Table 3. ReLU and LeakyReLU achieve 80.1% and 80.0% AP_{50}, respectively, slightly outperforming GELU (79.7%), SiLU (79.3%), and ELU (79.1%). This suggests that activation functions with sharper transitions create more discriminative importance patterns. All activation functions significantly outperform the baseline ScoreNet (78.0%), confirming that AWP's core insight is robust to this design choice.

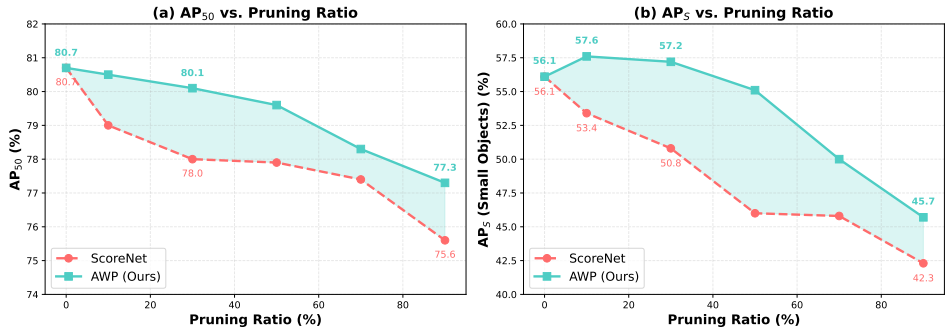

Figure 3: **Pruning-ratio comparison on SparseFormer.** AWP (solid) outperforms ScoreNet (dashed). Gains at $k = 0.7$: $+2.1$ AP_{50} and $+6.4$ AP_S points; at $k = 0.5$: $+9.1$ AP_S points.

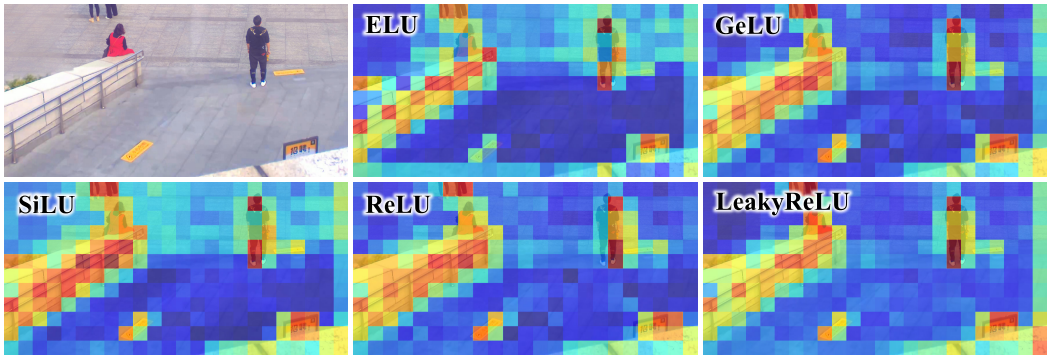

Figure 4: **Activation patterns across different activation functions.** FFN activation heatmaps on Swin Transformer for five activation functions on the same input image.

Aggregation Method. We compare four aggregation strategies in Table 5. Mean gives the highest AP_{50} (79.7%) and AP_S (55.0%), but slightly lower AP_L (83.3%) than min, max, or L2 (83.7–83.9%). The largest gains occur on small objects.

Table 5: **Aggregation method comparison.** Results use $k = 0.7$, GELU, and 12 fine-tuning epochs. Mean aggregation performs best overall, especially on small objects.

Aggregation	AP_{50}	AP_S	AP_M	AP_L
Mean	**79.7**	**55.0**	**78.8**	83.3
Min	78.5	43.5	76.3	**83.9**
L2 Norm	78.3	41.3	76.4	83.7
Max	78.3	41.4	76.8	83.7
ScoreNet (baseline)	78.0	50.8	78.1	82.3

5 Conclusion

We presented AWP, a parameter-free window selection method that uses intrinsic FFN activations to identify informative regions in high-resolution images. On PANDA, it achieves state-of-the-art detection with substantially reduced FLOPs, and the activation signal transfers across four backbones, with quantitative detection results on PANDA and COCO and activation analyses on aerial imagery. A training-free polarity check accommodates differences between local and pooled global pathways, enabling practical deployment without a learned scoring module. The same intrinsic score supports zero-shot deployment and optional detector fine-tuning, providing a direct route from spatial activations to efficient regional computation.

Acknowledgments and Disclosure of Funding

This work was supported in part by the National Natural Science Foundation of China (NSFC; Nos. 82441013 and 62572268); the National Science and Technology Major Project (No. 2022ZD0119402); the Ministry of Science and Technology of China (contract No. 2024YFB2809103); the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM904); the "Pioneer" and "Leading Goose" R&D Program of Zhejiang (No. 2024C01142); and the Tsinghua University – Migu Xinkong Culture Technology (Xiamen) Co., Ltd. Joint Research Center for Intelligent Light Field and Interaction Technology, Tsinghua University - Joint R&D Project (R2511DKW) – Phase 2. We also acknowledge the KLATASDS-MOE.

References

[1] 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. *CoRR*, abs/2103.14030, 2021. URL https://arxiv.org/abs/2103.14030.

[2] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

[3] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013. URL http://arxiv.org/abs/1311.2524.

[4] Ross Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015. doi: 10.1109/ICCV.2015.169.

[5] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL http://arxiv.org/abs/1506.01497.

[6] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. URL http://arxiv.org/abs/1506.02640.

[7] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018. URL http://arxiv.org/abs/1804.02767.

[8] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. YOLOX: exceeding YOLO series in 2021. *CoRR*, abs/2107.08430, 2021. URL https://arxiv.org/abs/2107.08430.

[9] Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Yolov10: Real-time end-to-end object detection, 2024. URL https://arxiv.org/abs/2405.14458.

[10] Yunjie Tian, Qixiang Ye, and David Doermann. Yolov12: Attention-centric real-time object detectors, 2025. URL https://arxiv.org/abs/2502.12524.

[11] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020. URL https://arxiv.org/abs/2005.12872.

[12] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M. Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection, 2022. URL https://arxiv.org/abs/2203.03605.

[13] Xueyang Wang, Xiya Zhang, Yinheng Zhu, Yuchen Guo, Xiaoyun Yuan, Liuyu Xiang, Zerun Wang, Guiguang Ding, David Brady, Qionghai Dai, and Lu Fang. Panda: A gigapixel-level human-centric video dataset. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3265–3275, 2020. doi: 10.1109/CVPR42600.2020.00333.

[14] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Dota: A large-scale dataset for object detection in aerial images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3974–3983, 2018.

[15] Wenxi Li, Yuchen Guo, Jilai Zheng, Haozhe Lin, Chao Ma, Lu Fang, and Xiaokang Yang. Sparseformer: Detecting objects in hrw shots via sparse vision transformer. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 4851–4860, 2024.

[16] Haozhe Lin, Chunyu Wei, Li He, Yuchen Guo, Yunqi Zhao, Shanglong Li, and Lu Fang. Gigatraj: Predicting long-term trajectories of hundreds of pedestrians in gigapixel complex scenes. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19331–19340, 2024. doi: 10.1109/CVPR52733.2024.01829.

[17] M. Tao, Bing Bai, Haozhe Lin, Heyuan Wang, Yu Wang, Lin Luo, and Lu Fang. When visual grounding meets gigapixel-level large-scale scenes: Benchmark and approach. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22119–22128, 2024. doi: 10.1109/CVPR52733.2024.02088.

[18] Xuanyao Chen, Zhijian Liu, Haotian Tang, Li Yi, Hang Zhao, and Song Han. Sparsevit: Revisiting activation sparsity for efficient high-resolution vision transformer, 2023. URL `https://arxiv.org/abs/2303.17605`.

[19] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[20] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html, 2010.

[21] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018.

[22] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016.

[23] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017. URL `http://arxiv.org/abs/1708.02002`.

[24] Fatih Cagatay Akyon, Sinan Onur Altinuc, and Alptekin Temizel. Slicing aided hyper inference and fine-tuning for small object detection. *2022 IEEE International Conference on Image Processing (ICIP)*, pages 966–970, 2022. doi: 10.1109/ICIP46576.2022.9897990.

[25] Bharat Singh, Mahyar Najibi, and Larry S Davis. Sniper: Efficient multi-scale training. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper_files/paper/2018/file/166cee72e93a992007a89b39eb29628b-Paper.pdf`.

[26] Jiahao Fan, Huabin Liu, Wenjie Yang, John See, Aixin Zhang, and Weiyao Lin. Speed up object detection on gigapixel-level images with patch arrangement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4653–4661, June 2022.

[27] Wenxi Li, Ruxin Zhang, Haozhe Lin, Yuchen Guo, Chao Ma, and Xiaokang Yang. Saccadedet: A novel dual-stage architecture for rapid and accurate detection in gigapixel images, 2024. URL `https://arxiv.org/abs/2407.17956`.

[28] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. URL `https://arxiv.org/abs/2010.11929`.

[29] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *CoRR*, abs/2012.12877, 2020. URL https://arxiv.org/abs/2012.12877.

[30] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer v2: Scaling up capacity and resolution. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11999–12009, 2022. doi: 10.1109/CVPR52688.2022.01170.

[31] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster, 2023. URL https://arxiv.org/abs/2210.09461.

[32] Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=BjyvwnXXVn_.

[33] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *CoRR*, abs/2106.02034, 2021. URL https://arxiv.org/abs/2106.02034.

[34] Lingchen Meng, Hengduo Li, Bor-Chun Chen, Shiyi Lan, Zuxuan Wu, Yu-Gang Jiang, and Ser-Nam Lim. Adavit: Adaptive vision transformers for efficient image recognition. *CoRR*, abs/2111.15668, 2021. URL https://arxiv.org/abs/2111.15668.

[35] Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Wei Niu, Mengshu Sun, Xuan Shen, Geng Yuan, Bin Ren, Hao Tang, et al. Spvit: Enabling faster vision transformers via latency-aware soft token pruning. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XI*, pages 620–640. Springer, 2022.

[36] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *CoRR*, abs/2102.12122, 2021. URL https://arxiv.org/abs/2102.12122.

[37] Hongxu Yin, Arash Vahdat, Jose M. Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-vit: Adaptive tokens for efficient vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10809–10818, June 2022.

[38] Lin Song, Songyang Zhang, Songtao Liu, Zeming Li, Xuming He, Hongbin Sun, Jian Sun, and Nanning Zheng. Dynamic grained encoder for vision transformers. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 5770–5783. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/2d969e2cee8cfa07ce7ca0bb13c7a36d-Paper.pdf.

[39] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Molchanov_Importance_Estimation_for_Neural_Network_Pruning_CVPR_2019_paper.html.

[40] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Mmdetection: Open mmlab detection toolbox and benchmark. *CoRR*, abs/1906.07155, 2019. URL http://arxiv.org/abs/1906.07155.

[41] Fan Yang, Heng Fan, Peng Chu, Erik Blasch, and Haibin Ling. Clustered object detection in aerial images. *CoRR*, abs/1904.08008, 2019. URL http://arxiv.org/abs/1904.08008.

[42] Kai Chen, Zerun Wang, Xueyang Wang, Dahan Gong, Longlong Yu, Yuchen Guo, and Guiguang Ding. Towards real-time object detection in gigapixel-level video. *Neurocomputing*, 477: 14–24, 2022. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2021.12.049. URL `https://www.sciencedirect.com/science/article/pii/S0925231221018877`.

[43] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvtv2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, 8(3):1–10, 2022.

[44] Qihang Fan, Huaibo Huang, Mingrui Chen, Hongmin Liu, and Ran He. Rmt: Retentive networks meet vision transformers. In *CVPR*, 2024.

[45] Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. VisionZip: Longer is better but not necessary in vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19792–19802, 2025.

[46] Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. DivPrune: Diversity-based visual token pruning for large multimodal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9392–9401, 2025.

[47] Gong Cheng, Junwei Han, Peicheng Zhou, and Lei Guo. Multi-class geospatial object detection and geographic image classification based on collection of part detectors. *ISPRS Journal of Photogrammetry and Remote Sensing*, 98:119–132, 2014.

A Additional Ablation Studies

A.1 Ablation on Scoring Mechanisms

We conduct ablation experiments to quantify the computational advantages of our activation-based scoring mechanism compared to the learned ScoreNet module from SparseFormer [15]. ScoreNet predicts window importance using an MLP that processes residual features at each stage, requiring additional forward passes and learned parameters. In contrast, AWP directly leverages FFN activations that are already computed during the forward pass, requiring only lightweight operations (mean aggregation and normalization) without any learned components.

Table 6 presents a direct comparison between AWP and ScoreNet across three key metrics: detection accuracy (AP_{50}), computational cost (GFLOPs), and scoring latency. The results demonstrate that AWP not only achieves higher accuracy but also significantly reduces computational overhead.

Table 6: **Comparison of AWP vs. ScoreNet scoring mechanisms.** Both methods are evaluated on SparseFormer at $k = 0.7$ after fine-tuning. Latency is measured as the average scoring overhead across four stages over 100 forward passes of the backbone on a single RTX 4090 GPU. GFLOPs are calculated on processing a 1280×800 window. AWP achieves higher accuracy (+2.1 AP_{50} points), lower computational cost (-1.7 GFLOPs), and dramatically reduced latency (-84%).

Scoring Mechanism	AP_{50}	GFLOPs	Latency (ms)
AWP (Ours)	**80.1**	**74.0**	**0.063**
ScoreNet (Baseline)	78.0	75.7	0.397

Analysis. The results reveal three key advantages of AWP over ScoreNet:

(1) Higher Accuracy: AWP achieves 80.1% AP_{50}, outperforming ScoreNet by 2.1 points. This improvement demonstrates that FFN activation patterns provide a more reliable importance signal than learned variance-based scoring, particularly for identifying object-containing windows in gigapixel images.

(2) Lower Computational Cost: By eliminating the ScoreNet module, AWP reduces computational cost from 75.7 GFLOPs to 74.0 GFLOPs, saving approximately 1.7 GFLOPs. This reduction comes from removing the MLP network and its associated forward passes at each stage, while AWP only requires lightweight operations on already-computed FFN activations.

(3) Dramatically Reduced Latency: AWP achieves remarkably low scoring latency at only 0.063ms, representing an **84% reduction** compared to ScoreNet's 0.397ms. This dramatic speedup stems from AWP's parameter-free design—it only requires reading out already-computed FFN activations (negligible overhead), computing the mean, and normalization, without any matrix multiplications or learned parameters. In contrast, ScoreNet requires forward passes through an MLP network at each of the four stages, incurring substantial overhead.

Conclusion: AWP demonstrates a superior balance between efficiency and accuracy, achieving higher detection performance with lower computational cost and latency. This makes AWP particularly suitable for real-time gigapixel object detection applications where both accuracy and inference speed are critical.

B Implementation and Training Details

B.1 Model Architecture

SparseFormer Backbone Configuration. Our SparseFormer backbone follows a hierarchical design with four stages. The detailed configuration is presented in Table 7. Each stage contains both Global Attention blocks for coarse-grained feature aggregation and Local Attention blocks for fine-grained processing on selected windows. The embedding dimensions progressively increase from 96 to 768 across stages, following the standard hierarchical vision transformer design.

AWP Integration. For AWP, we read out the FFN intermediate activations at each stage during the forward pass. The captured activations are aggregated using mean pooling over tokens within each

Stage	Embed Dims	Num Heads	Depth	MLP Ratio
0	96	3	2	4
1	192	6	2	4
2	384	12	6	4
3	768	24	2	4

Table 7: **SparseFormer model configuration details.** The architecture follows a hierarchical design with progressively increasing embedding dimensions and attention heads. Window size is set to 7×7 across all stages. MLP ratio of 4 means the FFN hidden dimension is $4\times$ embed_dims.

window, producing a scalar importance score per window. These scores are then normalized and used for top-k window selection. Importantly, AWP introduces **zero additional parameters**—it only requires lightweight operations (mean, normalization, top-k selection) on already-computed FFN activations.

DINO Detection Head. We use DINO [12] as the detection head with 900 queries, following the standard DETR-based detection pipeline. The encoder and decoder each contain 6 layers with 256 embedding dimensions. The neck uses a ChannelMapper to project multi-scale features from the backbone (192, 384, 768 channels) to a unified 256 channels.

B.2 Training Configurations

Hardware and Batch Size. We train all models on 8 NVIDIA RTX 4090 GPUs with a total batch size of 16 (2 samples per GPU). All latency measurements reported in Table 6 are conducted on a single RTX 4090 GPU.

Optimizer and Learning Rate. The training adopts the AdamW optimizer with an initial learning rate of 1×10^{-4} for most components and 1×10^{-5} for the backbone (using a learning rate multiplier of 0.1). Weight decay is set to 1×10^{-4} with gradient clipping at a maximum norm of 0.1.

Learning Rate Schedule. The learning rate follows a multi-step schedule with decay milestones at epochs 24 and 33, where the learning rate is reduced by a factor of 0.1. All models are trained for 36 epochs from scratch without ImageNet pretraining, following the standard practice for gigapixel detection benchmarks. For AWP fine-tuning experiments, we initialize from trained detector checkpoints and update the full backbone, neck, and detection head for an additional 6–8 epochs with AWP enabled. Table 1 uses 8 epochs; the AWP scoring rule has no learned parameters.

Data Augmentation. Following SparseFormer's training protocol, we apply random horizontal flipping with probability 0.5. For multi-scale training, we use two augmentation strategies: (1) direct random resize to scales ranging from $(480, 1333)$ to $(800, 1333)$, and (2) random resize to $(400, 4200)$ to $(600, 4200)$ followed by random cropping to $(384, 600)$ and then resizing to the target scales. Images are divided into multi-scale grids of 16×16, 8×8, 4×4, and 2×2 during training, with slices containing no objects removed.

Activation Function. For AWP experiments, we use ReLU as the activation function in FFN layers unless otherwise specified. As shown in the main paper, ReLU and LeakyReLU achieve the best performance, though all tested activation functions (GELU, SiLU, ELU) significantly outperform the learned ScoreNet baseline.

C Zero-shot Generalization Visualization on COCO

To demonstrate the generalization capability and transferability of AWP, we conduct zero-shot visualization experiments on the COCO dataset using an official pretrained Swin Transformer model without any fine-tuning. This experiment validates that the activation-based importance patterns emerge naturally in pretrained models and can be directly leveraged by AWP without additional training.

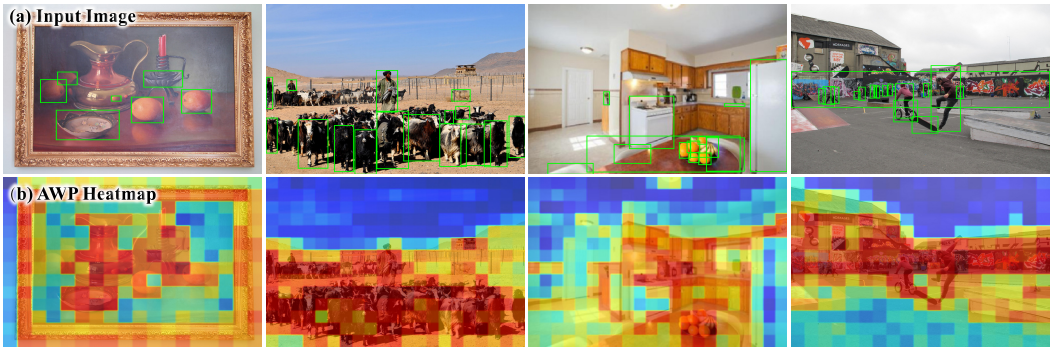

Figure 5: **Zero-shot activation visualization on COCO dataset.** We apply AWP to an official pretrained Swin-Tiny model without any fine-tuning. (a) Input images with ground truth bounding boxes (green). (b) Window-level FFN activation heatmaps, where red indicates high importance (foreground objects) and blue indicates low importance (background). The activation patterns clearly distinguish object-containing regions from empty backgrounds, demonstrating AWP's zero-shot generalization capability and cross-dataset transferability. All results are obtained using the official COCO-pretrained checkpoint without any modification.

C.1 Experimental Setup

We use the official Swin-Tiny model pretrained on COCO for object detection (Mask R-CNN with Swin-T backbone). The model is loaded directly from the official checkpoint without any modification or fine-tuning. We capture FFN activations from Stage 1 (the second stage) of the Swin Transformer backbone and compute window-level importance scores using mean aggregation, following the same procedure as our main experiments.

C.2 Qualitative Results

Figure 5 presents activation heatmaps on four representative images from the COCO validation set. The visualization clearly demonstrates that:

(1) Foreground-Background Distinction: Object-containing regions (e.g., paintings, people, kitchen appliances, motorcycles) consistently exhibit **higher activation values** (shown in red/yellow), while background regions (e.g., walls, sky, empty spaces) show **lower activation values** (shown in blue). This pattern emerges naturally without any explicit supervision for window importance.

(2) Cross-Dataset Transferability: Despite being trained on COCO (natural images with diverse objects), the activation patterns align well with semantic importance. This demonstrates that AWP's core principle—FFN activations encode spatial importance—generalizes across different datasets and image characteristics.

(3) Architecture Independence: The consistent activation patterns in Swin Transformer (a general-purpose backbone) validate that AWP is not specific to SparseFormer or gigapixel detection. The method can be applied to any window-based vision transformer without architectural modifications.

(4) Zero-shot Capability: Most importantly, these results are obtained **without any fine-tuning or retraining**. AWP can be directly applied to pretrained models by simply capturing and aggregating FFN activations, enabling immediate deployment and performance gains.

C.3 Multi-Backbone Results on COCO

To complement the Swin-only visualization above and to support the multi-backbone discussion in Section 4.4, we provide both the qualitative heatmaps (Figure 6) and the full per-keep-ratio AP table (Table 8) on COCO across three different backbones (Swin-T, PVTv2-B0, and RMT-S). Across all three backbones, FFN activations align with semantically important object regions, confirming that the AWP signal transfers to denser detection scenes beyond gigapixel imagery and is not specific to a single architecture family.

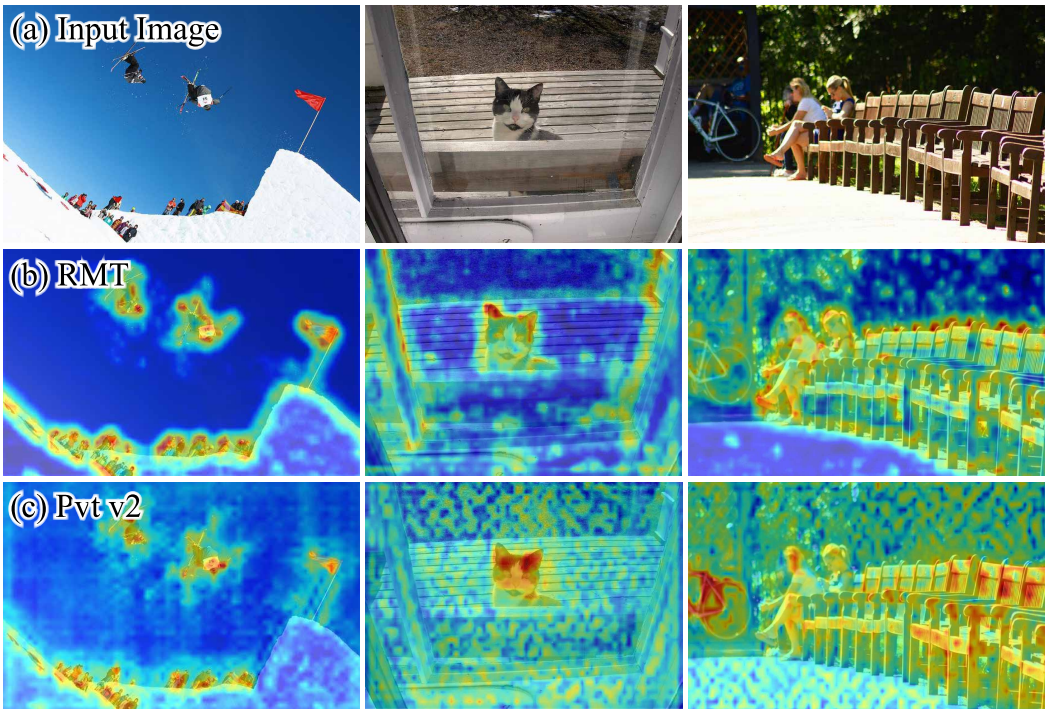

Figure 6: **Zero-shot AWP heatmaps on COCO** for Swin-T, PVTv2-B0, and RMT-S. Across all three backbones, FFN activations align with semantically important object regions, indicating that the AWP signal transfers to denser detection scenes beyond gigapixel imagery.

Table 8: **Zero-shot AWP on COCO across backbones and keep ratios.** AWP is inserted into publicly available pretrained detectors (Swin-T, PVTv2-B0, RMT-S) without any retraining. Even on a dense detection benchmark, AWP remains effective across all three backbones with moderate degradation at medium keep ratios.

Keep ratio	Swin-T	PVTv2-B0	RMT-S
1.0	42.70	37.10	49.00
0.9	42.35	36.05	48.52
0.7	41.90	34.95	46.58
0.5	40.20	34.10	43.22
0.3	38.00	32.80	40.83
0.1	35.01	27.83	39.04

C.4 Discussion

These zero-shot visualization results suggest two practical takeaways. First, the activation patterns persist on a pretrained checkpoint with no fine-tuning, so AWP can be plugged into existing models for immediate deployment. Second, the patterns are visible on multiple general-purpose backbones trained on natural images, indicating that the FFN-activation signal is not tied to gigapixel-specific architectures.

D Additional Visualizations on DOTA

To further test cross-dataset transferability, we additionally apply AWP zero-shot to a Swin-T detector on DOTA-v1.0 [14], an aerial object-detection benchmark in the HRW family. Figure 7 presents activation heatmaps on representative aerial scenes. Despite the substantial domain shift (aerial top-down imagery vs. the close-up natural images Swin was pretrained on), AWP's FFN-activation

heatmaps still concentrate on object-containing regions, indicating that the same window-importance signal also emerges on aerial data.

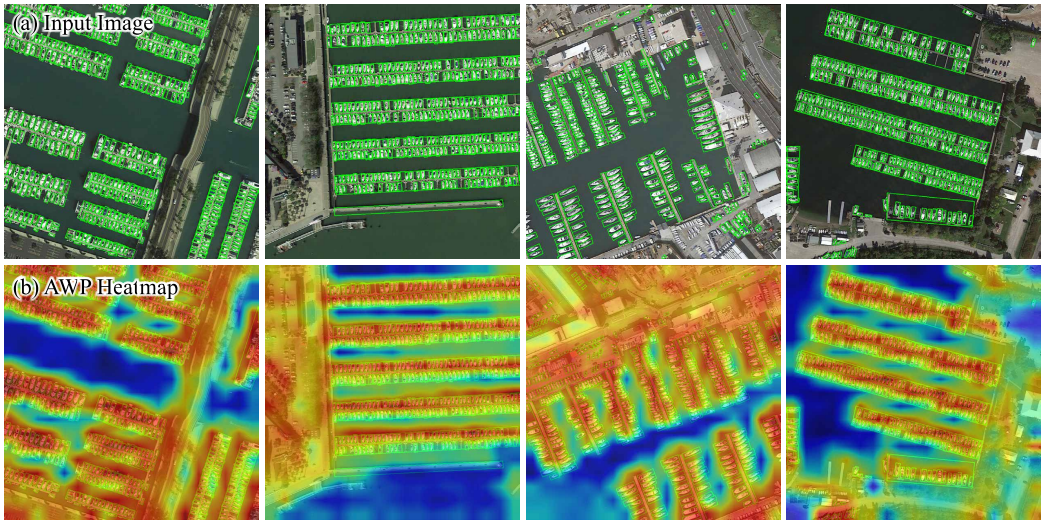

Figure 7: **Zero-shot AWP visualization on DOTA-v1.0 with Swin-T.** Top: input images with ground-truth boxes; bottom: window-level FFN activation heatmaps. The pattern observed on PANDA and COCO holds on aerial imagery as well.

E A Controlled Experiment on Polarity

In the main paper we report that, among the four vision transformer backbones we tested, SparseFormer is the only backbone for which AWP ranks in *ascending* order: foreground windows produce *lower* FFN activations rather than higher ones. Since SparseFormer's main architectural difference from Swin Transformer is its pooled global block, a natural hypothesis is that this polarity flip is driven by the structural role of the pooled global pathway, rather than by the backbone identity itself.

Setup. To test this hypothesis directly, we transplanted a SparseFormer-style pooled global branch into a standard Swin Transformer backbone. Specifically, we preserved the original local window blocks of Swin and inserted a global branch that average-pools each window into a single token, processes them through a lightweight global block, and fuses the output back into the token stream before subsequent local processing. We then fine-tuned this modified detector on PANDA for 4 epochs. The result is a single backbone that contains both a local window FFN and a pooled global FFN within the same network and the same training objective.

Result. Figure 8 visualizes FFN activation heatmaps for both pathways on the same input image. The original local window FFN retains the typical Swin Transformer pattern (foreground windows show *higher* activations), while the transplanted pooled global FFN exhibits the SparseFormer pattern (foreground windows show *lower* activations). Because both pathways live inside the same backbone and are trained with the same objective, this branch-dependent polarity flip cannot be attributed to a difference in detector, dataset, or training recipe. It points instead to the structural role of the pooled global pathway as the underlying cause of the opposite polarity observed in SparseFormer.

This controlled experiment supports the structural explanation used in the main paper: the AWP ranking signal itself is consistent across architectures, while its sign depends on whether the FFN reads full-resolution local tokens or pooled window-level tokens. Resolving the resulting binary choice (ascending vs. descending) is the role of the lightweight calibration procedure described in Appendix F.

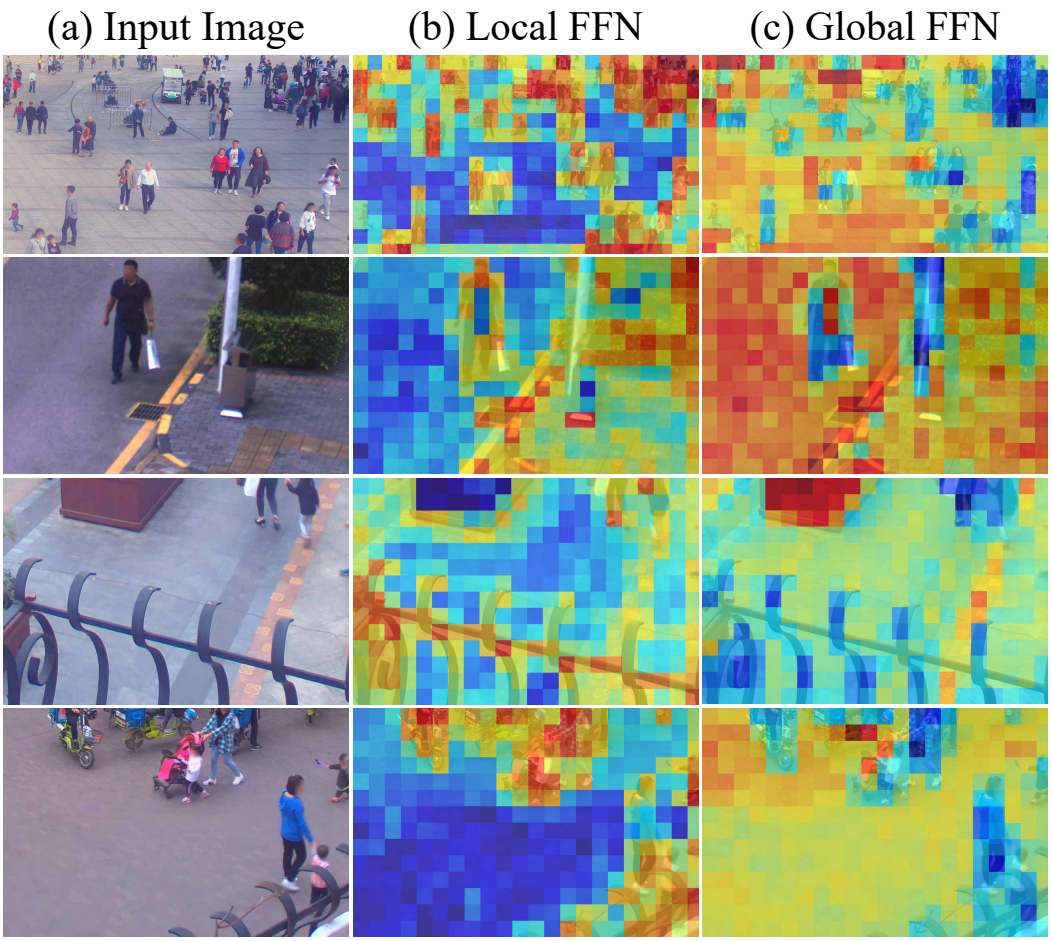

Figure 8: **Branch-dependent polarity flip in a Swin backbone with a transplanted global branch.** Within the same backbone trained on PANDA for 4 epochs, the original local window FFN exhibits the typical Swin Transformer polarity (foreground windows have higher activations), while the transplanted SparseFormer-style pooled global FFN exhibits the opposite polarity (foreground windows have lower activations). This isolates the pooled global pathway as the structural factor responsible for the polarity flip.

E.1 Window Perturbation Diagnostic

We also measured the effect of zeroing individual windows in a backbone stage output on the detector's neck features. Table 9 reports the L_2 perturbations for object-containing and object-free windows. For a direct ranking test, we sampled 40 random stage-1 windows per image on 5 PANDA images (200 windows total), masked each window separately, and measured relative neck-feature L_2 perturbation. Spearman correlation with the paper-defined absolute-mean FFN score was 0.749 for Swin and -0.384 for SparseFormer; reversing the latter ranking gives 0.384. Both correlations had $p < 3 \times 10^{-8}$, and the correlation sign agreed on all 5 images for each backbone.

Table 9: **Neck-feature perturbation from masking stage-output windows.** FG denotes object-containing windows and BG object-free windows. Values are L_2 changes in neck features.

Stage	Swin FG	Swin BG	SparseFormer FG	SparseFormer BG
1	127.0	122.0	120.9	124.4
2	116.9	115.6	109.3	113.7
3	130.9	104.0	123.4	127.5

F Zero-shot Polarity Calibration

As discussed in the main paper, for most vision transformer backbones we tested (Swin Transformer, PVTv2, RMT) foreground windows produce *higher* FFN activations than background windows, so AWP keeps the top-k scores in descending order. SparseFormer is the only exception in our study: its global block ingests average-pooled window tokens and exhibits the opposite pattern, so AWP must rank in ascending order. To remove the need to set this direction manually, we use a simple training-free polarity calibration procedure.

Procedure. For a given backbone and target dataset, we run the dense (un-pruned) detector on up to 10 randomly sampled unlabeled images (10 in the reported evaluation) to obtain the reference neck features F^{dense}. We then run two AWP-pruned variants at a fixed keep ratio (we use k=0.5): one ranking in *descending* order ($F^{\downarrow}$) and one in *ascending* order ($F^{\uparrow}$). For each image we compute the cosine similarity between each pruned variant and the dense reference, and select the polarity whose pruned features are closer to the dense ones. The full procedure is summarized in Algorithm 1.

Algorithm 1 Zero-shot Polarity Calibration

Require: detector $\mathcal{D}$ with backbone, unlabeled calibration set $\mathcal{X}$ ($|\mathcal{X}| \leq 10$), keep ratio k

$v_{\downarrow} \leftarrow 0,\ v_{\uparrow} \leftarrow 0$
for $x \in \mathcal{X}$ **do**
  $F^{\text{dense}} \leftarrow \text{Neck}(\mathcal{D}(x))$ *// no pruning*
  $F^{\downarrow} \leftarrow \text{Neck}(\mathcal{D}_{\text{AWP},\downarrow,k}(x))$ *// keep top-k highest*
  $F^{\uparrow} \leftarrow \text{Neck}(\mathcal{D}_{\text{AWP},\uparrow,k}(x))$ *// keep top-k lowest*
  $s_{\downarrow} \leftarrow \cos(F^{\downarrow}, F^{\text{dense}}),\ s_{\uparrow} \leftarrow \cos(F^{\uparrow}, F^{\text{dense}})$
  $v_{\downarrow} \mathrel{+}= [s_{\downarrow} > s_{\uparrow}],\ v_{\uparrow} \mathrel{+}= [s_{\uparrow} > s_{\downarrow}]$
end for
return descending if $v_{\downarrow} > v_{\uparrow}$ else **ascending**

Empirical Results. Table 10 reports the calibration outcome on four backbones using 10 randomly sampled images. For each backbone we also report the average cosine-similarity margin ΔSim, defined as the per-image gap between the correct and the incorrect polarity ($\Delta\text{Sim} > 0$ means the correct polarity wins). The procedure recovers the correct polarity on $\geq 9/10$ images for every backbone, with consistently positive margins, confirming that the polarity can be set automatically and reliably from a handful of unlabeled images.

Table 10: **Polarity calibration on four backbones at k=0.5.** "Selected polarity" is the direction returned by Algorithm 1; the fraction in parentheses is the per-image agreement with that polarity over 10 randomly sampled calibration images. ΔSim is the average cosine-similarity gap (correct $-$ incorrect). SparseFormer is the only backbone that calibrates to ascending order.

Backbone	Dataset	Selected polarity	Avg. ΔSim
Swin-T [1]	PANDA	Descending (10/10)	+0.0149
SparseFormer-T [15]	PANDA	Ascending (9/10)	+0.0093
PVTv2-B0 [43]	COCO	Descending (10/10)	+0.0203
RMT-S [44]	COCO	Descending (10/10)	+0.0075

Sensitivity to calibration-image choice. We exhaustively evaluated all subsets of sizes $n \in \{1, 3, 5, 7, 9\}$ from each ten-image calibration set using Algorithm 1. For Swin-T, PVTv2-B0, and RMT-S, every subset recovered the direction in Table 10. For SparseFormer-T, 90% of one-image subsets and 100% of subsets of every tested larger size recovered that direction. Thus every three-image subset succeeded on all four backbones; single-image calibration succeeded in 39 of the 40 cases overall.

G End-to-End Efficiency

We measure the complete FP32 detector forward pass, including the backbone, neck, and DINO head, on one NVIDIA RTX 3090. Inputs are 1280×800 patches processed in a true batch of 16. Timing uses `torch.cuda.synchronize()`, and peak-memory statistics are reset before each measurement. Table 11 includes activation capture and sparse execution. Per-patch latency is batch latency divided by 16; throughput counts processed patches per second. The $\pm$ values describe variation across timing iterations.

Table 11: Full-detector efficiency on RTX 3090, FP32, batch 16.

Configuration	Batch (ms)	ms/patch	Patches/s	Peak memory (MB)
Dense ($k = 1.0$)	1176.662 ± 2.952	73.541	13.598	7543.5
ScoreNet ($k = 0.7$)	1054.464 ± 2.285	65.904	15.174	6503.1
AWP ($k = 0.7$)	1045.190 ± 3.354	65.324	15.308	6502.8
AWP ($k = 0.5$)	967.815 ± 2.288	60.488	16.532	5903.5

Relative to dense execution, AWP at $k = 0.7$ reduces latency by 11.2% and peak memory by 13.8%, while increasing throughput by 12.6%. At $k = 0.5$, the corresponding improvements are 17.8%, 21.7%, and 21.6%. At the same $k = 0.7$, AWP improves throughput over ScoreNet by approximately 0.9%, with essentially equal peak memory. Measurements at batch sizes 1, 3, and 8 preserve the same ordering.

Activation capture. AWP reads post-nonlinearity FFN activations produced during the detector forward pass and reduces them over channels and window positions to obtain one score per window. Mean activation-capture time is below 0.1 ms at both tested keep ratios. These read/store operations are included in Table 11; the isolated scoring measurements in Table 6 use a separate RTX 4090 setup.

H Robustness and Additional Selector Comparisons

H.1 Variability across PANDA Scenes

Table 12 reports mean AP_{50} and sample standard deviation across PANDA's five independent large scenes. All methods use the same SparseFormer checkpoint and zero-shot evaluation protocol. This statistic describes scene-to-scene variation under a fixed checkpoint. AWP has the highest mean at each keep ratio, exceeding the better alternative by 0.9, 0.6, and 0.5 points at $k = 0.9$, 0.7, and 0.5, respectively.

Table 12: PANDA AP_{50} across five scenes (mean $\pm$ sample standard deviation).

Keep ratio	AWP	SparseViT-style	ToMe-style
0.9	79.4 ± 0.55	78.5 ± 0.34	78.2 ± 0.60
0.7	78.6 ± 0.61	78.0 ± 1.20	77.8 ± 0.88
0.5	77.6 ± 0.45	77.0 ± 1.47	77.1 ± 1.10

H.2 Zero-shot Selector Comparison on Swin-T

We compare AWP with ToMe [31], VisionZip [45], and DivPrune [46] adapted to the same Swin-T detector, checkpoint, evaluation, and nominal keep ratios (Table 13). AWP exceeds the strongest alternative by 2.2 AP_{50} points at $k = 0.7$ and 2.0 points at $k = 0.5$.

Table 13: Zero-shot selector comparison on Swin-T, PANDA AP_{50}. Dense Swin-T reaches 75.0.

Selector	$k = 0.7$	$k = 0.5$
ToMe	70.8	69.5
VisionZip	70.2	69.0
DivPrune	70.5	69.8
AWP	**73.0**	**71.8**

Backbone behavior. SparseFormer is trained for sparse execution, so replacing ScoreNet changes the selection rule within its existing execution regime. Swin-T is trained densely; inserting AWP changes both window selection and the computation performed by subsequent blocks. At $k = 0.5$, eight-epoch detector fine-tuning raises Swin-T AP_{50} from 71.8 to 72.5, recovering part of the gap to its dense 75.0 baseline. The SparseFormer comparison measures selector replacement, while the Swin experiment also measures adaptation from dense to sparse execution.

H.3 Activation Separation on Aerial Images

Using a COCO-pretrained Swin detector without retraining, we measure activation scores on 50 NWPU VHR-10 images [47]. Mean scores are 0.1588 for object-containing windows and 0.1542 for object-free windows. The per-image difference is 0.00443 ± 0.00177, and its sign is positive on all 50 images. These measurements extend the activation analysis to aerial scenes.

I Scoring Interpretation and Implementation Details

I.1 Spatial Activations and Attention Maps

Post-nonlinearity FFN responses preserve the spatial token layout, allowing direct aggregation over each region. Backbone attention maps encode pairwise routing across tokens and heads; DINO's object queries represent multiple candidate objects in the decoder. Forming a single window score from these attention maps requires reductions across heads, queries, layers, or feature levels. AWP reads a scalar magnitude directly from the spatial FFN response. The Taylor approximation in Section 3.4 motivates aggregate magnitude, while Appendix E measures how the resulting ranking relates to downstream feature perturbation.

I.2 Regional Execution and Window Size

AWP uses a trained two-dimensional token grid and a regional gather/process/scatter path. Here "window" specifies the pruning unit. Swin uses local shifted-window attention, PVTv2 uses spatial-reduction attention over shared downsampled keys and values, and RMT uses Manhattan attention with distance-dependent spatial decay. All three expose spatial activations that can be aggregated regionally. Applying the same rule to a flat global-attention ViT requires token gather/scatter and recomputes attention over the retained token set.

The window side length is $M = 7$ for SparseFormer, Swin, PVTv2, and RMT. This retains the chosen sparse/window execution granularity across datasets. Ranking polarity is calibrated independently: local full-resolution FFNs use descending scores in the tested models, whereas SparseFormer's pooled global branch uses ascending scores. The hybrid-branch experiment in Appendix E isolates this difference within one detector.

I.3 Zero-shot Insertion and Detector Fine-tuning

Zero-shot evaluation inserts AWP into an already trained detector and keeps its weights fixed. Polarity calibration compares dense and pruned neck features on up to ten unlabeled images. Optional fine-tuning updates the backbone, neck, and detection head to adapt to the selected sparse execution pattern. The score remains the absolute activation mean throughout. Table 1 distinguishes zero-shot insertion from eight-epoch fine-tuning, while Appendix B gives the optimizer and training configuration.

I.4 Targets, Scene Content, and Metric Conventions

"Object-containing" and "object-free" refer to the annotated target categories: pedestrians on PANDA and the benchmark categories on COCO or aerial datasets. Foreground/background labels in the activation visualizations use this annotation-based meaning. AWP ranks the detector's task-trained response; image texture and visual depth are separate properties. HRW denotes a high-resolution image with a wide field of view, and AWP accepts different aspect ratios. "Gigapixel" describes the spatial resolution of an individual source frame; inference operates on its image patches.

AP_{50} is average precision at IoU 0.50. COCO AP averages over thresholds from 0.50 to 0.95 in steps of 0.05. All AP values are reported on a 0–100 scale. An absolute gain is the difference in AP points; retention is the ratio to the corresponding dense baseline. For example, zero-shot Swin-T retains $73.0/75.0 = 97.3\%$ at $k = 0.7$ and $71.8/75.0 = 95.7\%$ at $k = 0.5$.

Against SparseFormer at $k = 0.7$ (78.0 AP_{50}, 50.8 AP_S), fine-tuned AWP gains 2.1 and 6.4 points at $k = 0.7$, or 1.6 and 4.3 points at $k = 0.5$. The 9.1-point small-object gain in Figure 3 uses a different comparison: AWP and ScoreNet both at $k = 0.5$ (55.1 versus 46.0). Keeping these baseline choices explicit separates accuracy gains from relative performance retention.

J Ethical Statement

In compliance with ethical guidelines for computer vision research involving human subjects, we declare the following privacy protection measures implemented in this work:

Face Anonymization. All human faces appearing in the gigapixel images used for visualization and qualitative analysis in this paper have been systematically blurred or anonymized to protect individual privacy. This includes all figures showing PANDA scenes in the main paper and supplementary material. The anonymization process was applied consistently across all visual materials while preserving the scientific integrity of our experimental visualizations.

Dataset Compliance. Our experiments use publicly available PANDA [13] and COCO [2] data, with additional activation analyses on DOTA [14] and NWPU VHR-10 [47]. We adhere to all usage terms and conditions specified by the dataset creators.

Intended Applications. AWP improves computational efficiency for object detection, with applications in traffic monitoring, urban planning, and environmental analysis.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 clearly state our main claims (FFN activations naturally encode window importance; AWP is parameter-free and supports zero-shot use; quantitative gains on PANDA and generalization to Swin/PVTv2/RMT and to dense COCO detection), and these claims are directly supported by the experiments in Section 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the main limitations across multiple sections: (i) AWP requires a sufficiently trained backbone for the FFN signal to emerge, mentioned in Section 3; (ii) the ranking polarity is architecture-dependent (descending for Swin/PVTv2/RMT, ascending for SparseFormer), addressed by the calibration procedure in Appendix F and the controlled mechanism analysis in Appendix E; (iii) the accuracy changes with keep ratio are quantified in Section 4.4 and Table 8; Appendix I specifies the trained spatial representation and regional execution interface.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include formal theoretical results requiring proofs; AWP is an empirical method with the equations in Section 3 serving as the precise definition of the scoring procedure rather than as theorems.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The AWP procedure is fully specified by Eq. (4)–(5) in Section 3 and Algorithm 1 in Appendix F. Section 4 and Appendix B describe datasets, backbones, the DINO detection head, optimizer, learning-rate schedule, training/fine-tuning epochs, batch size, hardware, and data augmentation. All evaluated backbones (SparseFormer, Swin, PVTv2, RMT) and benchmarks (PANDA, COCO, DOTA) are publicly available.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Code is not released at submission time, but the AWP procedure is fully specified by Eq. (4)–(5), Algorithm 1, and the implementation details in Appendix B. All datasets (PANDA, COCO, DOTA) and backbone implementations (SparseFormer, Swin, PVTv2, RMT, MMDetection) are publicly available. We plan to release a clean reference implementation upon acceptance.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 reports datasets, evaluation metrics, the DINO detection head, the multi-scale slicing strategy, and the training/fine-tuning schedules. Appendix B provides further details: full backbone configuration, optimizer (AdamW with 1×10^{-4} learning rate, 0.1 backbone multiplier, weight decay 1×10^{-4}), gradient clipping, learning-rate milestones (epochs 24 and 33), data augmentation, and AWP hook placement.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Table 12 reports mean AP_{50} and sample standard deviation across five PANDA scenes for the fixed-checkpoint selector comparison. Appendix E reports Spearman correlations and p-values for the 200-window perturbation analysis. Appendix F enumerates calibration-image subsets. The scene statistics characterize variation across evaluation scenes.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix B reports that all training is performed on 8 NVIDIA RTX 4090 GPUs with a total batch size of 16 (2 samples per GPU), the standard schedule of 36 epochs from scratch, and additional 6–8 epochs for AWP fine-tuning. Scoring-latency measurements (Table 6) use one RTX 4090. Appendix G reports FP32 full-detector timing and peak memory on one RTX 3090 at batch 16.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: We have reviewed the NeurIPS Code of Ethics and the research complies in every respect. We use only publicly released datasets (PANDA, COCO, DOTA) under their original terms of use; all human faces appearing in PANDA visualizations are systematically blurred or anonymized, as detailed in Appendix J. The work targets a methodological improvement in detection efficiency, not surveillance, identification, or other potentially sensitive applications.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The Broader Impact statement in the main paper and the discussion in Appendix J note that AWP is a methodological improvement to detection efficiency for high-resolution wide imagery. Positive impacts include reduced compute and energy use for tasks such as traffic monitoring and urban planning. We do not develop or release any new pretrained model or dataset; the underlying gigapixel imagery (PANDA) could in principle be misused for surveillance, but our contribution is solely an efficiency-improving plug-in and is not specific to any sensitive application.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release any new pretrained model, dataset, or generative system that carries a high risk of misuse. AWP is a parameter-free, training-free scoring procedure that operates on top of existing publicly available detectors and datasets.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets used (PANDA [13], COCO [2], DOTA [14]) and all backbones / detection frameworks built upon (SparseFormer [15], Swin Transformer [1], PVTv2 [43], RMT [44], DINO [12], MMDetection [40]) are publicly available and are properly cited in the paper. We adhere to the original terms of use of each asset.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The paper does not release any new dataset, model, or other asset. AWP is a parameter-free scoring procedure that runs on top of existing publicly available backbones and benchmarks.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.

- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve any crowdsourcing experiments or research with human subjects. All datasets used are publicly available benchmarks collected by the original dataset authors.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve any new data collection from human subjects, and no IRB review is required. We use only publicly released benchmark datasets under their original terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were used only for minor writing assistance and language editing, and do not constitute any part of the core methodology, experiments, or results. Per the NeurIPS LLM policy, no declaration is required in this case.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.

- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.