SpanFormer: Multi-Level Adaptive Sparsity for Object Detection in High-Resolution Wide Shots

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

[1]Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing, China
[2]Department of Automation, Tsinghua University, Beijing, China
[3]KLATASDS-MOE, School of Statistics, East China Normal University, Shanghai, 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

Object detection in high-resolution wide (HRW) shots, where a single image can span gigapixel resolutions and kilometre-scale fields of view, suffers from extreme foreground sparsity, dramatically varying foreground density from sparse to crowded scenes, and object clusters whose spatial extent ranges from dozens to thousands of pixels within the same dataset. State-of-the-art sparse vision transformers for gigapixel detection select a fixed top-k fraction of windows by hand-crafted variance scores, which over-promote textured but object-free background such as foliage and patterned facades, cannot adapt the keep ratio to the wide range of crowd density, and confine each kept window to a 7×7 receptive field that is far smaller than the spatial extent of many real-world object clusters. We propose **SpanFormer**, a sparse vision transformer that broadens the computational span at three complementary granularities: (i) *Prototype Routing* replaces variance with learnable foreground/background prototypes regularised by a balance constraint that prevents prototype collapse, widening the *semantic span* of the scoring criterion; (ii) *Dynamic Top-k* predicts a per-image keep ratio with a lightweight selector, making the *compute span* elastic so that crowded scenes receive proportionally more compute while near-empty scenes are skipped; (iii) *Cluster Memory Attention* performs union-find clustering over the kept windows by joint cosine similarity and Chebyshev spatial radius and augments each window's K and V with cross-window memory drawn from its cluster, extending the *attention span* beyond the local window with all projection parameters shared with the local branch and therefore no added projection cost. On the PANDA gigapixel benchmark, SpanFormer improves AP_{50} from 78.0% to **80.3%** over the SparseFormer baseline while reducing backbone FLOPs by over **75%**.

*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).

1 Introduction

Advances in gigapixel imaging systems have made it possible to capture single shots that span square-kilometre fields of view at billion-pixel resolutions [1]. Object detection in such *high-resolution wide* (HRW) shots matters for surveillance, aerial inspection and crowd analytics [2, 3]. Yet the regime is qualitatively different from the close-up imagery that has driven detector design on benchmarks such as MS COCO [4]. Three challenges stand out. First, foreground is *extremely sparse*: instances of interest typically occupy less than 5% of the image, leaving the detector to search through a sea of background pixels. Second, the density of foreground varies *dramatically across scenes*, from a handful of isolated targets to crowded gatherings of thousands, a range that can span two orders of magnitude inside a single dataset. Third, objects span *vastly different scales* and form clusters tens to thousands of pixels wide, far beyond conventional window-based transformers' receptive fields.

A productive response to the first challenge has been to process HRW shots *sparsely*, applying expensive backbone computation only to windows that look informative [5–8]. The current state of the art on gigapixel benchmarks, exemplified by SparseFormer [5] and its MoE-augmented successor GigaMoE [9], share a common spatial-selection recipe: each non-overlapping window is reduced to a hand-crafted residual statistic (the high-frequency component obtained by subtracting the window's pooled mean) and scored by a single linear head, then a *fixed* top-k fraction is kept for the remainder of the backbone while the rest are replaced by a cheap global summary. Despite delivering large speed-ups, this recipe suffers from three concrete limitations that motivate our work. **(i)** Residual statistics conflate objectness with texture energy, so top-k budget flows to busy backgrounds (foliage, facades, roofs) alongside pedestrians; high-variance foreground worsens this by leaving many kept windows on clutter rather than missed targets. **(ii)** A globally fixed keep ratio is fundamentally mismatched with the per-image density variation noted above; the same k either discards real objects in crowded scenes or wastes compute on empty ones. **(iii)** Even correctly selected windows attend only within a 7×7 receptive field, far smaller than typical object clusters, preventing information exchange among nearby instances of the same class.

We argue that these limitations reflect one pattern: HRW detectors fix a *narrow computational span*: scoring uses low-level texture only, the keep budget is a global fraction, and attention stays inside one window. We therefore propose **SpanFormer**, a sparse vision transformer that *broadens its computational span at three complementary granularities*, each tackling one of the limitations above. (i) *Prototype Routing* replaces the residual-statistics scorer with a pair of learnable foreground/background prototypes, regularised by a balance constraint that prevents prototype collapse, widening the *semantic span* of the scoring criterion from low-level texture statistics to learned objectness. (ii) *Dynamic Top-k* predicts a per-image keep ratio with a lightweight selector supervised by the foreground proportion of the ground truth, making the *compute span* elastic so that crowded scenes receive proportionally more compute while near-empty ones are skipped almost entirely. (iii) *Cluster Memory Attention* (CMA) performs union-find clustering over the kept windows by joint cosine similarity and Chebyshev spatial radius, and augments each window's K and V with cross-window memory tokens drawn from its cluster, extending the *attention span* from a single 7×7 block to the full extent of a semantically related cluster. Crucially, CMA reuses the local branch's W_K, W_V projections, so this extra reach comes at *no additional projection cost*. Figure 1 illustrates these mechanisms on HRW patches.

We instantiate SpanFormer as a drop-in backbone for the DINO detector [10] and evaluate it on the PANDA gigapixel benchmark. SpanFormer improves AP_{50} from 78.0% to **80.3%** over the SparseFormer baseline while reducing backbone FLOPs by over **75%** (75.71 G $\rightarrow$ 17.53 G); it also surpasses the recent MoE-augmented GigaMoE by $+1.2$ $\mathbf{AP}_{50}$ at roughly **one third** of its backbone FLOPs. The added parameters, contributed by the prototypes, the dynamic-k selector and the cluster routine, together account for less than 1% of the backbone, so the gains come from a better use of compute rather than a larger model. Our contributions are:

- **Prototype Routing** for HRW window scoring. We replace hand-crafted variance with a small set of learnable foreground/background prototypes, regularised by a balance constraint that prevents collapse, so that windows are ranked by learned objectness rather than texture energy.
- **Dynamic Top-K** for image-adaptive sparsity. We train a lightweight per-image selector against the foreground proportion of the ground truth, replacing the globally fixed keep ratio

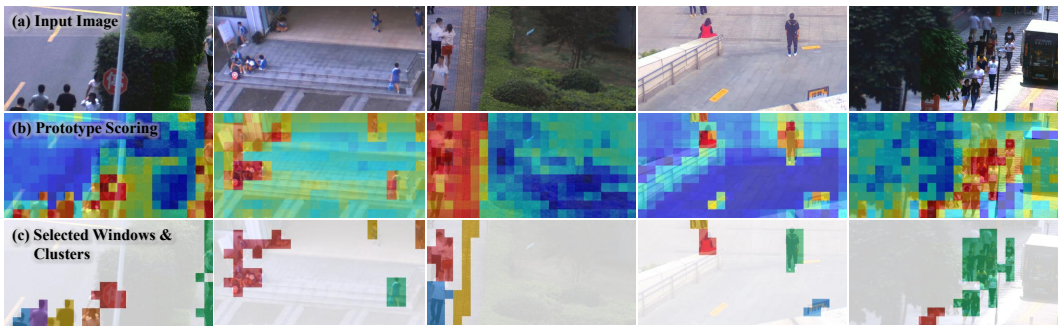

Figure 1: **SpanFormer broadens the computational span at three complementary granularities. (a) Input Image. (b) Prototype Scoring**: per-window similarity to our learned prototypes (the redder, the more similar to the foreground prototype; the bluer, the more similar to the background prototype), suppressing textured background such as foliage and patterned facades. **(c) Selected Windows & Clusters**: after Dynamic Top-k pruning (white = pruned), the kept windows are tinted by their union-find cluster id (Cluster Memory Attention).

of prior sparse detectors with a budget that contracts on near-empty scenes and expands on crowded ones.

- **Cluster Memory Attention (CMA)** for cross-window context. We perform on-the-fly union-find clustering of the kept windows by joint cosine similarity and Chebyshev radius, and let each window attend to representative tokens of its cluster, extending the receptive field beyond the 7×7 local block at *zero* added projection cost.

- Together these three modules establish a new accuracy–efficiency Pareto frontier on the PANDA gigapixel benchmark: SpanFormer improves AP_{50} by $+2.3$ over the SparseFormer baseline and by $+1.2$ over the recent GigaMoE, while cutting backbone FLOPs by more than 75% and 60% respectively.

2 Related Work

Sparse vision transformers for high-resolution detection. The challenge of detecting objects in gigapixel-level high-resolution wide (HRW) imagery has motivated a line of work that processes each image *sparsely*, restricting expensive backbone computation to windows that look informative. Early approaches couple a coarse cluster proposal stage with a fine-grained second-stage detector [7, 11, 12], or rely on cropping/slicing the input into uniform tiles [6, 8, 13, 14]. SparseFormer [5] replaces these heuristics with a lightweight per-window scorer inside a Swin-style backbone, retaining only a fixed top-k fraction of windows for the remaining attention layers. Its recent successor GigaMoE [9] keeps the same selection front-end but replaces the FFN with a Mixture-of-Experts module to add adaptivity inside each kept window. Their shared reliance on hand-crafted residual statistics, a fixed keep ratio, and a single-window receptive field motivates the three failure modes we address.

Adaptive computation in vision models. A parallel body of work seeks to spend compute non-uniformly across tokens or layers in a vision transformer. DynamicViT [15], EViT [16], SPViT [17] and SparseViT [18] prune unimportant tokens layer by layer with a learned scorer, while A-ViT [19] halts redundant tokens via early-exit. AdaViT [20] additionally selects which patches, heads and layers to execute per image, and DiffRate [21] makes the pruning ratio itself differentiable. Token merging [22] reduces sequence length without an extra scorer, and Dynamic-Head [23] applies similar ideas to detection necks; closer to our setting, DGE [24] learns a dynamic patch granularity for sparse backbones. Conditional computation has also been explored in width (MoE [25]) and language-side adaptive depth [26]. These methods, however, are validated on benchmarks (ImageNet, COCO) where the foreground proportion varies modestly across images, and they all commit to a global pruning schedule. Our Dynamic Top-k instead conditions the keep ratio explicitly on the foreground proportion of *each* HRW shot, where this quantity can swing by two orders of magnitude.

Cross-window and memory-augmented attention. Window-based attention famously trades long-range modelling for quadratic-cost reduction. Swin [27, 28], CSWin [29] and Focal Transformer [30] restore inter-window information flow through cyclic shifts, cross-shaped windows or expanded receptive fields, but the exchange is purely *geometric*: every window only ever talks to a fixed set of neighbours. In language models, a complementary line attaches an *external memory* of key/value pairs that any token can attend to: Set Transformer [31] introduces inducing points, the Compressive [32] and Memorizing Transformer [33] retrieve from past context, and kNN-LM/RETRO [34, 35] inject neighbours from an external corpus. Mixture of Attention Spans [36] further heterogenises the per-head sliding-window length to enlarge the effective context. Cluster Memory Attention can be seen as the 2D vision counterpart of these memory-augmented designs: rather than retrieving from a global memory bank or extending the geometric window, each kept window is linked at runtime to a small set of *semantically similar neighbours* discovered by union-find clustering, yielding an asymmetric Q / K,V routing that costs no extra projection parameter.

3 Method

We instantiate the three span-broadening modules introduced in Section 1 on top of the SparseFormer [5] backbone. Each backbone stage applies, in order, a window-level scoring step that produces top-k keep indices, followed by sparse local self-attention on the kept windows; SpanFormer replaces this scoring + selection + attention pipeline with our three light-weight modules while preserving the original Swin-style shifted-window backbone [27] so that windows excluded in one layer can re-enter the kept set in the next. Figure 2 shows the full pipeline. Throughout, $\mathbf{x}_i \in \mathbb{R}^{W^2 \times C}$ denotes the tokens of window i (window size $W{=}7$, channels C), and N is the number of windows in the stage. We obtain each representative $\bar{\mathbf{x}}_i \in \mathbb{R}^C$ by average-pooling $\mathbf{x}_i$ to one token, applying the stage's global self-attention to the N-token sequence, and then LayerNorm:

$$\bar{\mathbf{x}}_i \;=\; \mathrm{LN}\Big(\mathrm{GlobalAttn}\big(\{\mathrm{AvgPool}(\mathbf{x}_j)\}_{j=1}^{N}\big)\Big)_i, \tag{1}$$

which is the same global branch that SparseFormer [5] uses; we re-use $\bar{\mathbf{x}}_i$ throughout this section.

3.1 Prototype Routing

To replace the residual-statistics scorer of prior sparse detectors [5, 9], we attach a small set of learnable prototypes $\mathbf{P} = [\mathbf{p}_1; \ldots; \mathbf{p}_M] \in \mathbb{R}^{M \times C}$ to every backbone stage. The score of window i is the cosine similarity between its representative $\bar{\mathbf{x}}_i$ and each prototype:

$$s_{i,m} \;=\; \cos\big(\bar{\mathbf{x}}_i,\, \mathbf{p}_m\big), \qquad m = 1, \ldots, M. \tag{2}$$

Each stage uses a *foreground/background* prototype pair ($M{=}2$) to produce a logit per window:

$$\ell_i \;=\; \tau_s \cdot \big(s_{i,1} - s_{i,2}\big), \tag{3}$$

where τ_s is a fixed scaling constant (we use $\tau_s = 10$ throughout). Compared with the residual scorer of SparseFormer, ℓ_i measures *semantic* similarity to learned templates rather than the local energy of the window, which makes it robust against textured-but-empty regions such as foliage and patterned facades. Window logits are normalised across the stage by a softmax, $\sigma_i = \mathrm{softmax}_i(\ell_i)$, and feed the selection module (Sec. 3.2). We additionally reuse SparseFormer's global-feature gating

$$\mathbf{x}_i \;\leftarrow\; \mathbf{x}_i + (1 - \sigma_i)\, \bar{\mathbf{x}}_i, \tag{4}$$

so that pruned windows still receive a coarse update from the global branch.

Anti-collapse balance loss. Without explicit regularisation a pair of prototypes can collapse to a single mode (both vectors representing the same kind of background or foreground), in which case the foregroundness logit degenerates to noise. We therefore add a soft *balance constraint* on the expected per-prototype responsibility. With temperature T and per-window weights w_i given by the sparse-selection softmax,

$$q_m \;=\; \frac{\sum_i w_i \left[\mathrm{softmax}(\mathbf{s}_i / T)\right]_m}{\sum_i w_i}, \qquad \mathcal{L}_{\text{bal}} \;=\; \frac{1}{M} \sum_{m=1}^{M} \mathrm{ReLU}\big(\rho - q_m\big), \tag{5}$$

where ρ is the minimum allowed mass per prototype (we set $\rho{=}0.35$). The hinge form only penalises a prototype when its responsibility falls below ρ, leaving SpanFormer free to learn asymmetric foreground/background frequencies that match the data.

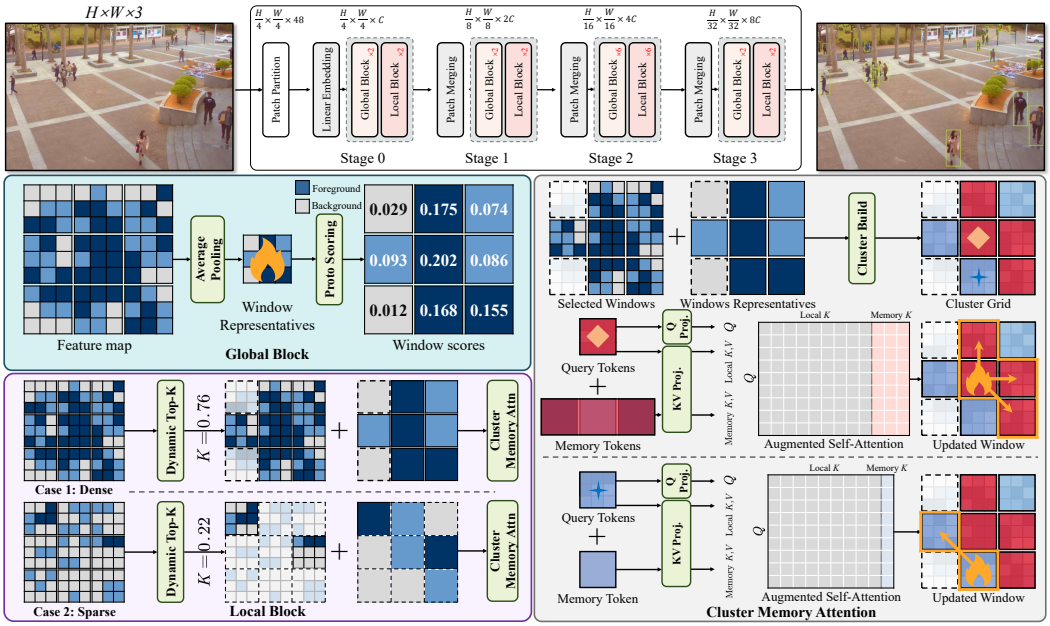

Figure 2: **SpanFormer pipeline. Top:** a 4-stage hierarchical backbone with Patch Partition, Linear Embedding and Patch Merging; SpanFormer modules sit inside every stage's *Global Block* and *Local Block* (the left panels below). **Global Block (left top):** per-window features are first average-pooled and then refreshed by global self-attention (the flame icon marks this attention update), producing *window representatives* $\bar{\mathbf{x}}_i$ that are scored against two learned prototypes (Blue = foreground-like, light = background-like). **Local Block (left bottom):** *Dynamic Top-k* sets a per-image keep ratio: k=0.76 in a dense scene, k=0.22 in a sparse one. **Cluster Memory Attention (right):** *Cluster Build* groups the kept windows by union-find on $\bar{\mathbf{x}}_i$; we mark one representative per cluster (diamond and 4-pointed star) and unfold its *Augmented Self-Attention*, where Query Tokens plus Memory Tokens are projected by shared W_K, W_V (and a window-only W_Q) and the right *Memory K, V* strip widens with cluster size. In the resulting *Updated Window*, the flame icon again denotes that the window has been refreshed by attention, and the yellow arrows trace the cluster siblings each query attends to.

3.2 Dynamic Top-k

Because the foreground proportion of HRW shots varies by orders of magnitude across scenes, no single keep ratio is appropriate for the entire dataset. We replace the fixed top-k schedule of SparseFormer/GigaMoE with a per-image *Dynamic Top-k* predictor that conditions the keep ratio r on the input itself. Concretely, we take the average-pooled per-window features $\{\mathrm{AvgPool}(\mathbf{x}_i)\}_{i=1}^{N}$ (the same input that the global self-attention later refines into $\bar{\mathbf{x}}_i$), arrange them on the stage's spatial grid, and apply a 3×3 adaptive average pool to produce a fixed-size descriptor $\mathbf{z} \in \mathbb{R}^{9C}$ regardless of the original grid size. This descriptor is fed to a two-layer MLP with sigmoid output:

$$r \;=\; \sigma\big(\mathrm{MLP}(\mathbf{z})\big) \in (0,1), \qquad k \;=\; \max\big(1,\, \min(\lfloor r \cdot N \rceil,\, \rho_{\max} N)\big), \tag{6}$$

where $\rho_{\max} \in (0,1]$ caps the worst-case budget (we use $\rho_{\max} = 1.0$). We retain the top k windows under the prototype softmax; the other $N - k$ contribute only to SparseFormer's global summary.

Foreground-fraction supervision. Training r end-to-end through the non-differentiable top-k operator is unstable. Instead, we expose r to a direct regression target derived from the ground-truth boxes. For every window in the stage's grid, we declare the window *positive* when either (i) the window's centre lies inside any GT box, or (ii) the window intersects any GT box with intersection-area-over-window-area $\geq \alpha$ (we use $\alpha = 0.1$). Letting $r^\star$ be the fraction of positive windows in the image, we add

$$\mathcal{L}_{\text{dynk}} \;=\; \lambda_{\text{dynk}} \cdot (r - r^\star)^2, \tag{7}$$

which fits the predicted ratio to the informative budget without a per-window classification loss.

Algorithm 1 Cluster Memory Attention (one stage)

Require: kept window indices $\mathcal{K}$, representatives $\{\bar{\mathbf{x}}_i\}_{i\in\mathcal{K}}$, tokens $\{\mathbf{x}_i\}_{i\in\mathcal{K}}$, thresholds τ, R, cap $M_{\max}$

compute pair-wise $s_{ij} = \cos(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j)$ and d_{ij} for all $i, j \in \mathcal{K}$
$\mathcal{E} \leftarrow \{(i,j) : s_{ij} > \tau \text{ AND } d_{ij} \le R\}$
$\mathrm{UnionFind}(\mathcal{E}) \rightarrow$ clusters $\{\mathcal{C}_c\}_c$
for all $i \in \mathcal{K}$ **do**
  $\mathcal{M}_i \leftarrow$ top-$M_{\max}$ representatives of $\mathcal{C}_{c(i)}$ ranked by s_{ij} $\{j \neq i\}$
  $\mathbf{K}_i, \mathbf{V}_i \leftarrow [\mathbf{x}_i;\, \mathcal{M}_i]\,\mathbf{W}_K, [\mathbf{x}_i;\, \mathcal{M}_i]\,\mathbf{W}_V$
  $\mathbf{Q}_i \leftarrow \mathbf{x}_i \mathbf{W}_Q$
  $\mathbf{x}_i \leftarrow \mathrm{softmax}(\mathbf{Q}_i \mathbf{K}_i^\top / \sqrt{d})\,\mathbf{V}_i$
end for

3.3 Cluster Memory Attention

Even after Prototype Routing and Dynamic Top-k have selected the right windows, each kept window's local self-attention is still confined to a $W \times W$ block, far smaller than the spatial extent of typical object clusters in HRW imagery. *Cluster Memory Attention* (CMA) extends local attention with a small *cross-window memory* from semantically similar windows.

On-the-fly cluster construction. For every kept window i we already have a representative $\bar{\mathbf{x}}_i$. We pre-compute, in parallel on the GPU,

$$s_{ij} \;=\; \cos\big(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j\big), \qquad d_{ij} \;=\; \max\big(|y_i - y_j|,\, |x_i - x_j|\big), \tag{8}$$

the pair-wise cosine similarity and the Chebyshev distance between the two windows on the stage's grid. We then keep an undirected edge (i, j) whenever

$$s_{ij} > \tau \quad \text{AND} \quad d_{ij} \le R, \tag{9}$$

and run a single-pass union-find [37] on the resulting edge set, producing a partition of the kept windows into clusters $\{\mathcal{C}_c\}_c$. The thresholds (τ, R) are stage hyper-parameters; we use τ=0.85 and R=2. The semantic constraint ($s_{ij} > \tau$) keeps the cluster homogeneous, while the spatial cap ($d_{ij} \le R$) discourages spurious long-range edges and keeps the cluster size compact, which we cap at $M_{\max}$ tokens per window (set per-stage to $\{16, 8, 4, 2\}$ from shallow to deep).

Asymmetric attention with shared K/V. Given clusters, CMA augments the local attention of each kept window i with a memory of representatives from its cluster siblings:

$$\mathcal{M}_i \;=\; \big\{\bar{\mathbf{x}}_j \;:\; j \in \mathcal{C}_{c(i)},\, j \neq i\big\}, \qquad |\mathcal{M}_i| \le M_{\max}. \tag{10}$$

For window i, we concatenate local tokens and memory to form the keys and values:

$$\mathbf{Q}_i = \mathbf{x}_i \mathbf{W}_Q, \quad \mathbf{K}_i = [\mathbf{x}_i;\, \mathcal{M}_i]\,\mathbf{W}_K, \quad \mathbf{V}_i = [\mathbf{x}_i;\, \mathcal{M}_i]\,\mathbf{W}_V, \tag{11}$$

$$\mathrm{CMA}(\mathbf{x}_i) = \mathrm{softmax}\Big(\mathbf{Q}_i \mathbf{K}_i^\top / \sqrt{d}\Big)\,\mathbf{V}_i \;\in\; \mathbb{R}^{W^2 \times d}. \tag{12}$$

Two design choices keep CMA *essentially free*. First, $\mathbf{W}_K, \mathbf{W}_V$ are shared between the local branch and the memory branch, so CMA introduces **no extra projection parameters**. Second, the routing is asymmetric: each representative $\bar{\mathbf{x}}_j$ contributes a single token to $\mathcal{M}_i$ rather than the full W^2 tokens of window j, so the per-query attention complexity grows from $\mathcal{O}(W^4)$ to $\mathcal{O}\big(W^2(W^2 + M_{\max})\big)$, which is essentially the same as local attention when $M_{\max} \ll W^2$. Algorithm 1 summarises one CMA pass.

3.4 Training Objective

We augment the DINO [10] detection objective with the two auxiliary losses:

$$\mathcal{L}_{\text{total}} \;=\; \mathcal{L}_{\text{DINO}} \;+\; \lambda_{\text{bal}} \cdot \sum_s \mathcal{L}_{\text{bal}}^{(s)} \;+\; \lambda_{\text{dynk}} \cdot \sum_s \mathcal{L}_{\text{dynk}}^{(s)}, \tag{13}$$

where s indexes backbone stages and we set λ_{bal}=0.1, λ_{dynk}=0.1 throughout. CMA itself contributes *no additional loss term*: its parameters are trained end-to-end through the detection objective, while the union-find clustering is parameter-free.

Method	Backbone	GFLOPs-F	GFLOPs-B	GFLOPs-A	AP_{total}	AP_S	AP_M	AP_L
General Two-Stage Detectors								
FasterRCNN [38]	ResNet-101	14.15	268.98	283.14	-	19.0	55.2	74.4
FasterRCNN* [38]	ResNet-50	10.35	196.71	207.07	70.5	20.3	71.2	76.0
RetinaNet [39]	ResNet-101	15.77	299.62	315.39	-	22.1	56.1	74.0
CascadeRCNN [40]	ResNet-101	15.54	295.24	310.78	-	22.7	57.9	76.5
Specialized Gigapixel Detectors								
ClusDet [11]	ResNet-50	10.35	196.71	207.07	71.8	21.9	69.6	78.2
DMNet [12]	ResNet-50	10.35	196.71	207.07	54.0	11.9	37.1	71.4
GigaDet [6]	CSP-DarkNet-53	4.61	87.59	92.20	68.4	21.0	59.9	76.2
PAN [7]	ResNet-50	10.35	196.71	207.07	71.5	25.6	71.9	76.8
Sparse Processing Methods								
Dynamic-Head [23]	Swin-T	5.74	109.1	114.8	59.2	16.5	53.7	69.4
Dynamic-Head+DGE [24]	PVT-DGE	6.12	60.11	66.23	57.5	15.4	50.8	69.5
Dynamic-Head+SparseFormer [5]	SparseFormer	6.29	58.35	64.64	77.1	36.4	74.0	**86.3**
Dynamic-Head+GigaMoE [9]	GigaMoE	6.07	46.87	52.94	77.9	37.7	74.3	85.5
Dynamic-Head+SpanFormer (Ours)	**SpanFormer**	**6.44**	**10.60**	**17.04**	**78.6**	**38.3**	**76.2**	85.7
DINO [10]	ResNet-50	6.21	118.02	124.24	54.2	28.9	53.0	59.2
DINO+SparseNet [5]	SparseNet	6.53	100.97	107.50	74.6	38.1	75.4	79.7
DINO [10]	Swin-T	6.64	126.19	132.84	60.6	36.7	61.2	64.9
DINO+DGE [24]	PVT-DGE	6.77	78.57	85.34	58.2	33.9	57.8	62.4
DINO+SparseFormer [5]	SparseFormer	6.90	68.81	75.71	78.0	50.8	78.1	82.3
DINO+GigaMoE [9]	GigaMoE	6.36	44.88	51.24	79.1	53.8	81.9	82.1
DINO+SpanFormer (Ours)	**SpanFormer**	**6.69**	**10.84**	**17.53**	**80.3**	**54.0**	**82.8**	**83.5**

Table 1: Comparison with the SoTAs on PANDA. Sparse-processing rows follow *Head+Backbone*: **DINO** denotes the DINO-head detector [10] and **Dynamic-Head** denotes Dynamic-Head [23], each paired with the backbone named in the adjacent column; remaining rows are standard two-stage or gigapixel-specialised setups. "F" and "B" denote foreground and background, respectively (A=F+B). AP is reported in percentage. Our SpanFormer results are highlighted in bold.

4 Experiments

4.1 Setup

Dataset and architecture. We evaluate on the PANDA gigapixel pedestrian-detection benchmark [1]; training images are tiled into 1280×800 patches and evaluation follows the standard COCO AP protocol. Unless noted, SpanFormer is plugged into DINO [10] (900 queries, 6+6 encoder/decoder layers) using a SparseFormer-Tiny backbone (depths $[2, 2, 6, 2]$, embedding 96, 7×7 windows) so that the detector head matches prior work; we additionally report Dynamic-Head [23] for completeness.

Training and FLOPs measurement. All models train for 12 epochs with AdamW (lr=1×10^{-4}, wd=10^{-4}, backbone $0.1\times$, step decay at epochs 8 and 11) at batch size 2 per GPU on $4\times$ A100/3090. The full set of method-specific hyperparameters (M, ρ, MLP shape, τ, R, $M_{\max}$, σ) is given in Appendix B. Because Dynamic Top-k makes the compute cost vary across the val set, we report the *mean per-image* backbone FLOPs measured by `mmengine.analysis`; for sparse-processing baselines [5, 9] we follow their original measurement protocol so that numbers are directly comparable.

4.2 Main Results

Table 1 compares SpanFormer with prior detectors on PANDA. With DINO, SpanFormer reaches **80.3%** AP_{50}, compared with 78.0% for SparseFormer and 79.1% for GigaMoE, while reducing backbone FLOPs from 75.71 G and 51.24 G to **17.53 G**. The trend also holds with Dynamic-Head (Figure 3). Appendices C–F provide DOTA results, token-reduction comparisons, keep-ratio validation, measured runtime, and complementary ablations.

4.3 Ablation Study

Module ablation. We isolate the contribution of each module by adding them one at a time on top of the SparseFormer baseline (Table 2a). Prototype Routing raises AP_{50} from 78.0 to 79.0, with backbone FLOPs decreasing from 75.71 to 57.82 G. Adding Dynamic Top-k reaches 79.4 AP_{50} at 21.33 G, reducing compute by 63.1% relative to the preceding variant. The full model with CMA achieves 80.3 AP_{50} at 17.53 G. Across these trained variants, the combined improvement over the baseline is 2.3 AP_{50} points with 76.8% fewer backbone FLOPs.

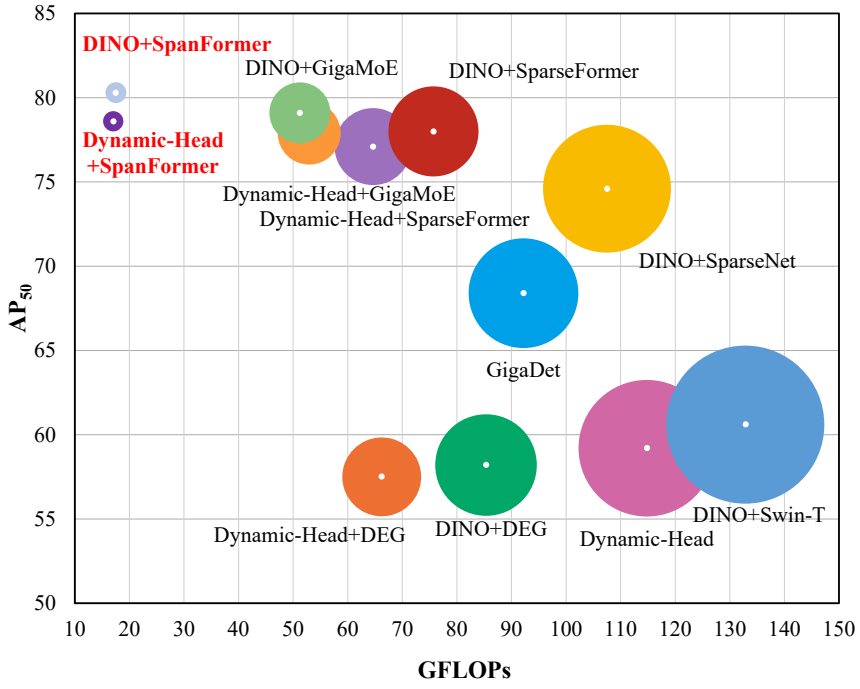

Figure 3: **Accuracy–efficiency Pareto on PANDA.** AP_{50} versus backbone GFLOPs; bubble area is proportional to backbone FLOPs. SpanFormer (red) sits alone in the upper-left, achieving the highest AP_{50} at the smallest compute among all sparse-processing detectors.

Variant	AP_{50}	GFLOPs
SparseFormer baseline	78.0	75.71
+ Prototype Routing	79.0	57.82
+ Dynamic Top-k	79.4	21.33
+ CMA (ours)	**80.3**	**17.53**

(a) Module-level ablation. Each row adds one module on top of the previous one.

Configuration	AP_{50}	GFLOPs
Full SpanFormer	**80.3**	**17.53**
w/o $\mathcal{L}_{\text{bal}}$	79.9	17.7
w/o $\mathcal{L}_{\text{dynk}}$	80.1	91.4
w/o both	79.5	92.0

(b) Auxiliary loss ablation. Without $\mathcal{L}_{\text{dynk}}$ the predicted keep ratio drifts toward its upper cap.

Table 2: Module-level (left) and auxiliary-loss (right) ablations on PANDA.

CMA hyperparameters. Table 3 reports four compact robustness studies, one per CMA hyperparameter, around our default operating point ($M_{\max}$=[16, 8, 4, 2], τ=0.85, R=2, σ=0.5, marked $\star$). AP_{50} ranges from 79.4 to 80.3 across the tested configurations. Every setting exceeds the 78.0 baseline, with at most 0.9 points separating it from the best result.

Auxiliary losses. We finally check whether the two auxiliary losses introduced in Sec. 3.4, namely the prototype balance loss $\mathcal{L}_{\text{bal}}$ and the dynamic-k supervision $\mathcal{L}_{\text{dynk}}$, both contribute to the final accuracy (Table 2b). Removing $\mathcal{L}_{\text{bal}}$ collapses the two prototypes to similar foreground-leaning vectors, so the foreground/background contrast disappears and AP_{50} drops by 0.4. Removing $\mathcal{L}_{\text{dynk}}$ leaves the per-image keep ratio unconstrained, so it drifts toward its upper cap during training and the backbone FLOPs balloon to 91 G (worse than the SparseFormer baseline) while AP_{50} stays essentially the same. Removing both yields a strong but costly variant resembling SparseFormer with prototype scoring (Table 2a).

4.4 Qualitative Analysis

Figure 4 compares four gigapixel patches. GigaMoE and SparseFormer allocate computation to textured backgrounds, including foliage, facades, and roofs. SpanFormer concentrates retained windows on pedestrians through foreground Prototype Routing and image-adaptive budgets.

$M_{\max}$ (per stage)	$\mathbf{AP}_{50}$	**GFLOPs**
$[8, 4, 2, 1]$	80.0	17.42
$[16, 8, 4, 2]$ $\star$	**80.3**	**17.53**
$[32, 16, 8, 4]$	79.9	18.38

R (strict)	$\mathbf{AP}_{50}$	**GFLOPs**
1	79.8	17.49
2 $\star$	**80.3**	**17.53**

τ	$\mathbf{AP}_{50}$	**GFLOPs**
0.80	80.2	16.60
0.85 $\star$	**80.3**	**17.53**
0.90	79.4	18.94
0.95	79.7	19.25

σ (per-head ratio)	$\mathbf{AP}_{50}$	**GFLOPs**
0.5 $\star$	**80.3**	**17.53**
0.75	79.5	18.74
1.0	79.7	18.88

Table 3: CMA hyperparameter robustness on PANDA. $\star$ marks our default; AP_{50} ranges from 79.4 to 80.3 across the tested settings.

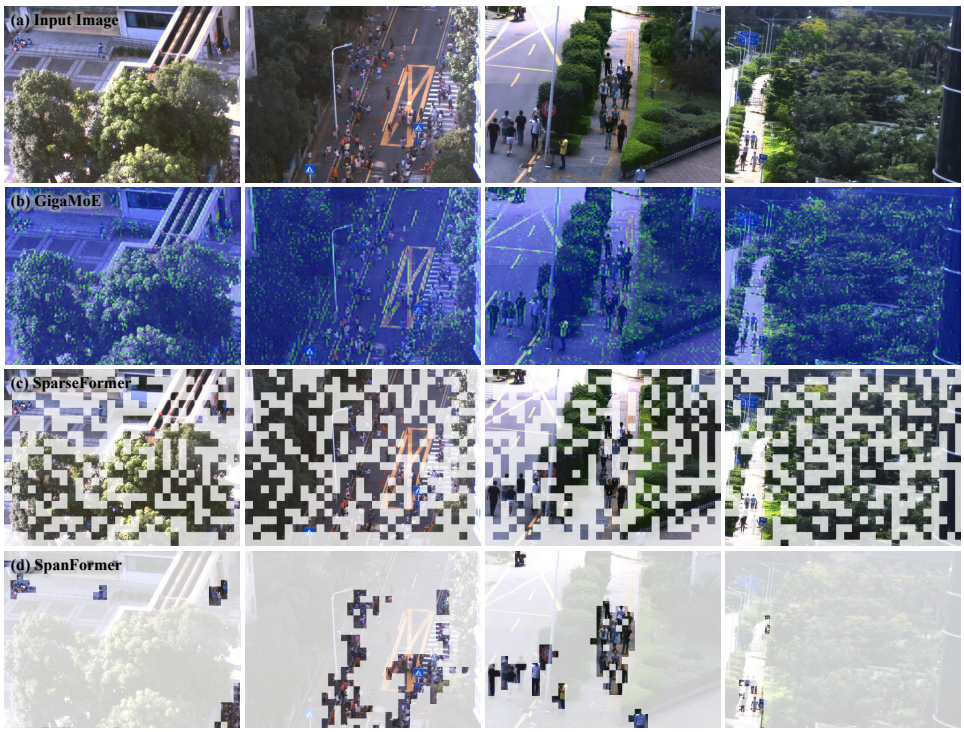

Figure 4: **Qualitative comparison on four gigapixel test patches.** **(a)** Input crop. **(b)** GigaMoE: expert routing density (greener = higher dispatched load). **(c)** SparseFormer: windows kept under fixed top-k=0.5. **(d)** SpanFormer: windows kept after Dynamic Top-k and Prototype Routing.

Practical efficiency. Measured runtime complements the FLOPs comparison. On an RTX 3090 at batch size 16, SpanFormer reduces backbone latency from 22.2 to 12.5 ms per patch (1.78× speedup) and peak batch memory from 5.67 to 5.05 GB. CMA adds 0.8 ms per patch to the variant without memory. For the complete DINO detector at batch size 3, latency decreases from 61.1 to 54.2 ms per patch (1.13× speedup); the unchanged detection head accounts for approximately 38 ms of the baseline runtime. Appendix E details the timing protocol and batch-size comparisons.

5 Conclusion

We presented **SpanFormer**, a sparse vision transformer that broadens the computational span of prior gigapixel detectors along three axes: a learned *semantic* span (Prototype Routing), an image-adaptive *compute* span (Dynamic Top-k), and a cluster-augmented *attention* span (Cluster Memory Attention). Together the three modules establish a new accuracy–efficiency Pareto on PANDA, lifting AP_{50} to 80.3% while cutting backbone FLOPs by more than 75% over the SparseFormer baseline.

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] 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.

[2] 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.

[3] 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.

[4] 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`.

[5] 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.

[6] 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`.

[7] 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.

[8] 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`.

[9] Xiang Li, Wenxi Li, Yuetong Wang, Chenyang Lyu, Haozhe Lin, Guiguang Ding, and Yuchen Guo. Gigamoe: Sparsity-guided mixture of experts for efficient gigapixel object detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(21):17553–17561, Mar. 2026. doi: 10.1609/aaai.v40i21.38810. URL `https://ojs.aaai.org/index.php/AAAI/article/view/38810`.

[10] 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`.

[11] 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.

[12] Changlin Li, Taojiannan Yang, Sijie Zhu, Chen Chen, and Shanyue Guan. Density map guided object detection in aerial images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[13] 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.

[14] 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.

[15] 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.

[16] 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_.

[17] 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.

[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] 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.

[20] 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.

[21] Mengzhao Chen, Wenqi Shao, Peng Xu, Mingbao Lin, Kaipeng Zhang, Fei Chao, Rongrong Ji, Yu Qiao, and Ping Luo. Diffrate: Differentiable compression rate for efficient vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 17164–17174, 2023.

[22] 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.

[23] Xiyang Dai, Yinpeng Chen, Bin Xiao, Dongdong Chen, Mengchen Liu, Lu Yuan, and Lei Zhang. Dynamic head: Unifying object detection heads with attentions. *CoRR*, abs/2106.08322, 2021. URL https://arxiv.org/abs/2106.08322.

[24] 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.

[25] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23 (120):1–39, 2022.

[26] Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33: 9782–9793, 2020.

[27] 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`.

[28] 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.

[29] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12124–12134, 2022.

[30] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal self-attention for local-global interactions in vision transformers. *arXiv preprint arXiv:2107.00641*, 2021.

[31] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019.

[32] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

[33] Yuhuai Wu, Markus N Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. *arXiv preprint arXiv:2203.08913*, 2022.

[34] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019.

[35] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR, 2022.

[36] Tianyu Fu, Haofeng Huang, Xuefei Ning, Genghan Zhang, Boju Chen, Tianqi Wu, Hongyi Wang, Zixiao Huang, Shiyao Li, Shengen Yan, et al. Mixture of attention spans: Optimizing llm inference efficiency with heterogeneous sliding-window lengths. *arXiv preprint arXiv:2406.14909*, 2024.

[37] Robert Endre Tarjan. Efficiency of a good but not linear set union algorithm. *J. ACM*, 22 (2):215–225, April 1975. ISSN 0004-5411. doi: 10.1145/321879.321884. URL `https://doi.org/10.1145/321879.321884`.

[38] 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`.

[39] 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`.

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

[41] 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.

[42] Jian Ding, Nan Xue, Yang Long, Gui-Song Xia, and Qikai Lu. Learning RoI transformer for oriented object detection in aerial images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[43] Wentong Li, Yijie Chen, Kaixuan Hu, and Jianke Zhu. Oriented RepPoints for aerial object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1829–1838, 2022.

[44] 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.

[45] 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.

A Ethical Considerations

SpanFormer is a backbone-side efficiency improvement for high-resolution wide (HRW) pedestrian detection. We highlight three considerations that follow from this scope.

Dataset privacy. Our pedestrian-detection experiments use the public PANDA gigapixel benchmark [1], which is released with the explicit intent of enabling academic research on crowd-scale detection. PANDA's authors apply face mosaicking on identifiable individuals before publication; all visualisations in this paper inherit that preprocessing, and no de-mosaicking is performed at any stage of our pipeline. We do not collect, scrape or annotate additional human imagery. The additional aerial-object experiments use the public DOTA-v1.0 benchmark.

Surveillance considerations. Pedestrian detectors at gigapixel scale could in principle be misused for surveillance. SpanFormer does not change the detector's output modality (it remains a generic person detector trained on a public academic benchmark), and our contribution is in compute allocation rather than in identification or tracking. Standard deployment mitigations – limiting downstream identification, restricting re-identification across cameras, and adhering to local privacy regulations – remain the responsibility of the deployer.

Energy footprint. A first-order positive impact of SpanFormer is energy efficiency: at matched accuracy it consumes roughly $4\times$ less backbone compute than the SparseFormer baseline, while measured inference speedups depend on batch size and whether the detection head is included (Appendix E).

B Implementation Hyperparameters

Table 4 summarises every method-specific hyperparameter used by the default SpanFormer in the main paper.

Module	Hyperparameter	Value
Prototype Routing	Number of prototypes per stage M	2 (foreground / background)
	Scaling constant τ_s	10
	Balance floor ρ	0.35
Dynamic Top-k	Adaptive-pool size	3×3
	MLP hidden width	64
	Worst-case cap $\rho_{\max}$	1.0
	GT area-fraction threshold α	0.10
Cluster Memory Attention	Similarity threshold τ	0.85
	Strict Chebyshev radius R	2
	Per-stage cap $M_{\max}$	$[16, 8, 4, 2]$
	Per-head share σ	0.5
Auxiliary loss weights	Balance loss λ_{bal}	0.1
	Dynamic-k loss λ_{dynk}	0.1

Table 4: Default SpanFormer hyperparameters used to obtain the main results in Table 1.

C Additional Benchmark and Token-Reduction Comparisons

C.1 Oriented Object Detection on DOTA

We evaluate SpanFormer on DOTA-v1.0 [41], which contains aerial imagery with 15 object categories and oriented bounding boxes. We follow the SparseFormer setup with RoI Transformer [42], replace its backbone with SpanFormer, and retain the fixed keep ratio of 0.7 for the SparseFormer baseline. Table 5 reports mAP at IoU 0.5 and GFLOPs. SpanFormer achieves 79.2 mAP, improving on SparseFormer by 1.26 points while reducing GFLOPs from 174.31 to 63.89 (63.3%). The fraction of baseline computation retained is 36.7% on DOTA, compared with 23.2% on PANDA, consistent with allocating more computation to denser imagery.

Table 5: Oriented detection on DOTA-v1.0.

Method	Backbone	mAP	GFLOPs
RoI Transformer [42]	ResNet-101	69.56	296.74
Oriented RepPoints [43]	Swin-T	77.63	221.32
RoI Transformer	SparseFormer	77.94	174.31
RoI Transformer	SpanFormer	**79.20**	**63.89**

C.2 Comparison with Token-Reduction Strategies

We adapt three token-reduction strategies to the same DINO-head and SparseFormer-Tiny pipeline on PANDA. VisionZip [44] and DivPrune [45] replace window selection with attention-centrality ranking and farthest-point sampling, respectively; ToMe [22] merges similar token pairs inside global blocks. Table 6 reports their detection accuracy across fixed keep ratios. These are adaptations to our detector, rather than evaluations of the original vision-language systems.

Table 6: Token-reduction strategies on PANDA (AP_{50}). The keep ratio is fixed for each baseline. SpanFormer predicts its ratio, with a realized mean of approximately 0.16.

Keep ratio	ToMe	VisionZip	DivPrune
0.1	–	75.2	76.9
0.3	–	75.9	77.9
0.5	78.9	76.0	78.2
0.7	78.0	76.7	78.4
SpanFormer (adaptive ratio)			**80.3**

The ToMe implementation evaluated here uses bipartite pair merging, so a single merging step removes at most half the input tokens; entries below 0.5 are not evaluated under this setup. SpanFormer exceeds the best baseline entry by 1.4 AP_{50} points. Its selection targets foreground content, while its image-dependent ratio allocates computation according to scene density. The realized keep ratio and GFLOPs measure different quantities: global processing and other fixed costs remain even when few windows undergo local processing.

D Direct Validation of Dynamic Top-k

D.1 Relation to Foreground Density

We compare the predicted keep ratio r with the ground-truth foreground-window fraction $r^\star$ on $3{,}285$ validation patches at all four stages ($13{,}140$ pairs). The target uses the same window-labeling criterion as training. Table 7 gives per-stage mean absolute error (MAE), Pearson correlation, and Spearman rank correlation. All reported correlations have $p < 10^{-75}$. The predicted ratio increases with foreground density, although its numerical range is narrower than that of $r^\star$.

Table 7: Predicted keep ratio versus ground-truth foreground fraction.

Stage	MAE	Pearson	Spearman
0	0.117	0.452	0.430
1	0.118	0.354	0.394
2	0.124	0.359	0.383
3	0.146	0.318	0.367

Grouping patches by foreground-density quartile yields monotonically increasing mean predictions at every stage. Table 8 shows stage 0: the mean prediction rises from 0.149 in near-empty patches to 0.205 in crowded patches. The predictor retains context in sparse scenes while devoting a larger budget to crowded scenes.

Table 8: Stage-0 predictions grouped by foreground-density quartile.

Density quartile	Mean $r^\star$	Mean predicted r
Q1 (near-empty)	0.000	0.149
Q2	0.012	0.162
Q3	0.091	0.178
Q4 (crowded)	0.266	0.205

D.2 Sensitivity to Keep-Ratio Errors

We perturb r at inference while preserving the prototype ranking. Table 9 reports changes in AP_{50} and realized keep ratio relative to the unperturbed model. Zero-mean random perturbations with relative standard deviation up to 30% change AP_{50} by at most 0.1 points; at 50%, accuracy decreases by 0.1 points. A systematic additive bias of -0.15 reduces the realized keep ratio to $0.24\times$ its original value and costs 1.8 AP_{50} points. A bias of $+0.15$ increases the ratio to $2.02\times$ and improves AP_{50} by 0.8 points. Thus the effect of a systematic error depends on its direction: underestimation sacrifices accuracy to save computation, whereas overestimation primarily increases computation.

Table 9: Inference-time perturbation of the predicted keep ratio. Random-noise levels are relative standard deviations; systematic biases are additive offsets.

Perturbation	ΔAP_{50} (points)	Relative keep ratio
Random, $\sigma = 10\%$	$+0.0$	$1.00\times$
Random, $\sigma = 20\%$	$+0.1$	$1.00\times$
Random, $\sigma = 30\%$	$+0.0$	$1.00\times$
Random, $\sigma = 50\%$	-0.1	$1.01\times$
Systematic bias -0.15	-1.8	$0.24\times$
Systematic bias $+0.15$	$+0.8$	$2.02\times$

E Measured Runtime and Memory

Measurement protocol. We measure trained checkpoints on one RTX 3090 using identical real evaluation patches, with preprocessing outside the timed region and one batch resident at a time. PANDA's 165 test frames produce 3,285 patches, with 16–35 patches per frame. The compared variants share the DINO head (900 queries and 6+6 encoder/decoder layers), SparseFormer-Tiny geometry, and 1280×800 patch size. Backbone timing includes routing, gathering, similarity computation, and union-find clustering. Latencies below are per patch, obtained by dividing batch latency by batch size; memory is the peak for the batch.

Table 10: Backbone runtime by batch size. Peak memory is rounded to 0.01 GB.

	Latency (ms/patch)				Peak memory (GB)	
Batch	Baseline	w/o CMA	SpanFormer	Speedup	Baseline	SpanFormer
3	23.6	13.3	14.9	1.58×	1.31	1.19
8	22.6	12.0	13.0	1.74×	2.99	2.68
16	22.2	11.7	12.5	1.78×	5.67	5.05

At batch 16, the backbone processes a batch in approximately 200 ms, compared with 355 ms for the baseline. Table 11 decomposes the change across trained variants: Prototype Routing has similar runtime to variance scoring, Dynamic Top-k produces the largest reduction, and CMA adds 0.8 ms per patch while restoring cross-window context.

Table 11: Module-wise backbone timing at batch 16. Each variant uses its own trained checkpoint.

Variant	Mean keep ratio	ms/patch	Peak memory (GB)
SparseFormer baseline	0.700	22.2	5.67
+ Prototype Routing	0.700	21.8	5.67
+ Dynamic Top-k	0.162	11.7	5.01
+ CMA (SpanFormer)	0.162	12.5	5.05

CMA overhead. The full CMA branch adds 1.6, 1.0, and 0.8 ms per patch at batch sizes 3, 8, and 16, respectively, and adds 8, 21, and 41 MB of peak memory. The similarity and union-find operations alone, summed across four stages, take 0.99, 0.35, and 0.18 ms per patch (Table 12). Their amortized cost decreases with batch size because clustering and gathering operate on batches of retained windows.

Table 12: Similarity computation and union-find clustering across four stages. These costs are included in the full backbone timings above.

Batch	Clustering (ms/patch)	Share of SpanFormer backbone
3	0.99	6.6%
8	0.35	2.7%
16	0.18	1.4%

Full detector runtime. Including the DINO detection head, latency at batch 3 decreases from 61.1 to 54.2 ms per patch, a 1.13× speedup. This gain is smaller than the backbone speedup because the unchanged detection head accounts for approximately 38 ms of the baseline runtime per patch.

F Complementary Module and Prototype Analyses

F.1 Leave-One-Out Module Ablation

Table 13 complements the sequential module ablation in the main paper by removing one component at a time from the full model. Removing Prototype Routing, Dynamic Top-k, or CMA decreases AP_{50} by 0.5, 1.2, or 0.9 points, respectively. Each component contributes when the other two are present. Together with the sequential ablation, these experiments cover six of the eight possible module combinations.

Table 13: Leave-one-out module ablation on PANDA. PR: Prototype Routing; DTK: Dynamic Top-k.

Variant	PR	DTK	CMA	AP_{50}
SparseFormer baseline	–	–	–	78.0
w/o Prototype Routing	–	✓	✓	79.8
w/o Dynamic Top-k	✓	–	✓	79.1
w/o CMA	✓	✓	–	79.4
Full SpanFormer	✓	✓	✓	**80.3**

F.2 Prototype Responsibilities during Training

We monitor the per-prototype responsibility q_m defined in Eq. (5), which measures the share of weighted window mass assigned to each prototype. Table 14 shows stage-0 values at four training iterations. Both prototypes retain substantial responsibility: the smaller share stays above 0.43, exceeding the balance-loss floor $\rho = 0.35$. The responsibilities gradually become asymmetric rather than converging to equal shares, as permitted by the hinge loss. These measurements complement the prototype-similarity visualization in Figure 1 and the balance-loss ablation in the main paper.

Table 14: Stage-0 prototype responsibilities during training ($q_1 + q_2 = 1$).

Iteration	q_1	q_2
50	0.542	0.458
9,200	0.546	0.454
18,400	0.559	0.441
36,700	0.564	0.436

F.3 Choice of Cluster-Based Memory

CMA groups retained windows when both semantic similarity and spatial proximity satisfy the edge criterion. Union-find produces a partition, so windows in a cluster share a consistent set of candidate memory representatives. The memory cap $M_{\max}$ is an upper bound: isolated windows receive no cross-window tokens, while larger groups can use more context. This differs from a fixed-k neighbor rule that assigns the same number of neighbors to every retained window. The spatial criterion also prevents direct links between distant but visually similar regions. The existing threshold and radius sweeps quantify sensitivity to the clustering rule; the leave-one-out result measures the benefit of adding cluster memory.

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 introduction list our three contributions (Prototype Routing, Dynamic Top-k, Cluster Memory Attention) and the main empirical claim ($+2.3$ AP_{50} over the SparseFormer baseline at over 75% FLOPs reduction); these match the results in Sec. 4.2 and the ablations in Sec. 4.3.

   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 limitations in Sec. 5 (single dataset, hyperparameter ranges studied), and the ablation in Table 2b explicitly shows that without the dynamic-k supervision the predicted keep ratio drifts toward its upper cap, indicating the practical importance of accurate ground-truth foreground statistics during training.

   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 theoretical results.

   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: Sec. 4.1 reports the dataset (PANDA), backbone configuration, optimizer, learning-rate schedule, batch size, and the full set of method-specific hyperparameters (Prototype Routing M, Dynamic Top-k MLP, CMA τ, R, $M_{\max}$, σ); Algorithm 1 fully specifies the cluster-construction procedure.

   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: [Yes]

   Justification: PANDA [1] is a public benchmark; we will release our SpanFormer source code and trained checkpoints upon acceptance, including configs that reproduce every row of Table 1 and the ablation tables.

   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: All training and evaluation details are reported in Sec. 4.1 and the hyperparameter sweeps in Sec. 4.3.

   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: [No]

   Justification: Following common practice in HRW gigapixel detection (e.g., SparseFormer, GigaMoE), we report single-run AP_{50} numbers because each training run takes several days on multi-GPU hardware. The hyperparameter robustness study in Table 3 (each axis

flat within ± 0.005 AP_{50} around the optimum) provides an empirical proxy for run-to-run variance.

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: Sec. 4.1 states that all models are trained for 12 epochs on $4\times$ NVIDIA A100/3090 GPUs with batch size 2 per GPU; per-image FLOPs are reported with mmengine.analysis (fvcore-based) at evaluation time.

   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: The research conforms to the NeurIPS Code of Ethics. We use only the publicly released PANDA benchmark and standard model components.

   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: SpanFormer is an efficiency-oriented detector for high-resolution wide imagery. Positive impacts include reducing the compute and energy required for gigapixel pedestrian detection, enabling deployment on more modest hardware; potential negative impacts inherit those of any pedestrian detector (e.g., misuse for surveillance), which we believe should be mitigated at the deployment level rather than restricted at the research level.

    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 a high-risk pre-trained model: SpanFormer is a backbone variant trained on the public PANDA benchmark for pedestrian detection, with no scraped data and no generative capability.

    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: PANDA [1], DINO [10], SparseFormer [5] and the other compared methods are properly cited; we use them under their respective public licenses.

    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: [Yes]

    Justification: The SpanFormer code and trained checkpoints will be released with the paper, accompanied by configuration files and instructions sufficient to reproduce all reported numbers.

    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 crowdsourcing nor research with human subjects.

    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 crowdsourcing nor research with human subjects.

    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 not used as any important, original, or non-standard component of the core methodology; their only use was for writing assistance (English polishing), which the NeurIPS LLM policy does not require declaring.

    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.