REPRESENTATIVE ATTENTION FOR VISION TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Many efficient attention methods reduce the quadratic cost of global self-attention by using predefined spatial partitions to construct compact proxies or restrict interactions. However, these partitions can separate distant tokens with similar features and group neighboring tokens with dissimilar features. This creates a mismatch between token interactions and the feature structure of visual content. To address this mismatch, we introduce *Representative Attention* (RPAttention), which organizes global communication through a compact set of input-dependent representatives. A learned projection of input token features determines their contributions to each representative. RPAttention then constructs representatives by GATHERING information from **ALL** image tokens rather than a predefined spatial subset. The constructed representatives then INTERACT through lightweight self-attention to refine the gathered content. They DISTRIBUTE their refined content to each spatial query using attention weights computed separately from the gathering assignments. With M representatives and N spatial tokens, the complete Gather–Interact–Distribute process requires $\mathcal{O}(NM + M^2)$ token interactions. Experiments on image classification, object detection, instance segmentation, and semantic segmentation demonstrate strong performance across multiple backbone families and competitive results against recent efficient attention methods. Runtime measurements further show favorable scaling as input resolution increases.

1 INTRODUCTION

Vision Transformers (ViTs) (Dosovitskiy et al., 2021) have become a standard architecture for visual representation learning, with self-attention enabling direct interactions between distant tokens. Visual tokens are typically obtained by dividing images into patches, and natural images contain substantial spatial redundancy (He et al., 2022). Explicit interactions between every pair of tokens may therefore be unnecessary for capturing global visual information. Nevertheless, standard global self-attention computes all pairwise interactions among N spatial tokens, incurring $\mathcal{O}(N^2)$ complexity. This quadratic cost becomes increasingly expensive at higher input resolutions, motivating more efficient attention mechanisms.

A central question is how to reduce these interactions without losing access to global information. Some methods restrict attention to local windows or neighborhoods (Liu et al., 2021; Dong et al., 2022; Hassani et al., 2023), while others retain a global communication path through compact proxies constructed using spatial reduction, pooling, or locally constrained associations (Wang et al., 2021; 2022; Han et al., 2024b; Huang et al., 2023). These approaches retain spatial priors in proxy construction or token association. This can separate distant tokens with similar features and combine neighboring tokens with dissimilar features. Spatial organization is useful for preserving visual detail, but it need not also determine how information is aggregated for global communication. We therefore ask: *can each representative aggregate information from the entire image based on token features rather than predefined spatial regions?*

We introduce *Representative Attention* (RPAttention) to address this question. Its core design separates how tokens contribute to global representatives from how spatial queries retrieve the refined content. Through a *Gather–Interact–Distribute* process, a learned projection first assigns each input token soft weights over representative anchors according to feature compatibility. The assignments are then normalized across tokens to aggregate their features into representatives. Each representative

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

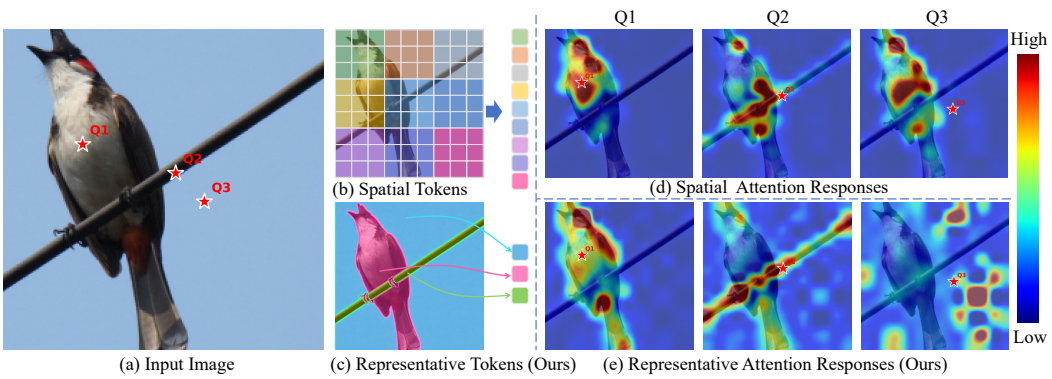

Figure 1: Motivation of RPAttention. (a) Input image with three query locations. (b,c) Spatial proxies follow predefined regions, whereas RPAttention gathers tokens globally by learned feature compatibility, allowing spatially distant tokens to contribute to the same representative. (d,e) Corresponding query-response maps.

can gather information from all image tokens rather than a predefined spatial subset, and its content depends on the current input. The representatives interact through self-attention, after which each spatial query retrieves their refined information through a separately computed attention map. As illustrated in Figure 1, spatially distant tokens can contribute to the same representative, while neighboring tokens need not share the same assignment. RPAttention preserves the spatial token layout and has a token interaction cost of $\mathcal{O}(NM + M^2)$ for M representatives.

Global communication without fixed spatial partitions is not unique to RPAttention. Kernel methods aggregate features into shared statistics (Katharopoulos et al., 2020; Choromanski et al., 2021; Han et al., 2023; Meng et al., 2025), while Slot Attention, Perceiver IO, and BiXT demonstrate the value of learned latent representations (Locatello et al., 2020; Jaegle et al., 2022; Hiller et al., 2024). Our contribution is a specific attention replacement that combines global competitive gathering, interaction among representatives, and independent query-conditioned distribution within standard vision backbones. We evaluate RPAttention across DeiT, PVT, PVTv2, and Swin on classification, detection, and segmentation tasks, together with analyses of representative behavior and practical efficiency. Our contributions are threefold:

- We propose organizing global communication through representatives constructed anew from image features at each layer, without predefined spatial groups or a separate latent sequence maintained across layers.
- We introduce RPAttention, an efficient global attention module built on learned representatives that can replace attention layers across standard vision backbones without changing the surrounding architecture.
- We demonstrate strong performance across four backbone families and three vision benchmarks, and characterize the behavior and practical efficiency of the representative bottleneck.

2 Related Work

Efficient Vision Attention. Efficient attention methods reduce the cost of dense token interactions through restricted connectivity, feature mappings, or compact approximations. Swin, CSWin, and NAT use windows or neighborhoods (Liu et al., 2021; Dong et al., 2022; Hassani et al., 2023), while DAT and BiFormer adapt sampling locations or select relevant regions from the input (Xia et al., 2022; Zhu et al., 2023). Random Wins All (Fan et al., 2026) instead uses fixed random token groups, showing that efficient communication need not follow a carefully designed spatial partition. Linear attention methods reorganize computation through feature mappings and shared statistics (Katharopoulos et al., 2020; Choromanski et al., 2021), with FLatten, PolaFormer, and InLine developing this direction for vision (Han et al., 2023; Meng et al., 2025; Han et al., 2024a). Linformer and Nyströmformer use low-rank projections or landmark approximations (Wang et al.,

2020; Xiong et al., 2021). RPAttention focuses on constructing explicit representatives from input features and using them to mediate global communication.

Spatial Proxy Construction. Several efficient vision transformers construct compact intermediate tokens using spatial structure. PVT and PVTv2 reduce the spatial resolution of keys and values (Wang et al., 2021; 2022). Agent Attention (Han et al., 2024b) constructs agent tokens by pooling query features, then uses them to aggregate global information and broadcast it back to spatial queries. STViT (Huang et al., 2023) initializes super tokens on a spatial grid and associates input tokens with nearby super tokens. These methods introduce spatial priors through proxy construction or local token association. RPAttention instead computes competitive assignments from token features to learned representative anchors and aggregates over all spatial locations. Its distinction lies in how representatives are constructed, rather than global information access alone.

Learned Aggregation and Latent Attention. Learning compact representations from visual features has been explored in several forms. TokenLearner (Ryoo et al., 2021) generates adaptive tokens through learned spatial weighting. Slot Attention (Locatello et al., 2020) uses competition across slots and normalized aggregation, followed by iterative slot refinement for object-centric representation learning. GroupViT (Xu et al., 2022) and kMaX-DeepLab (Yu et al., 2022) also learn grouping mechanisms for visual understanding and segmentation. RPAttention shares the principle of feature-dependent aggregation, but computes its assignments through a learned projection in a single gathering step per block. Its representatives support global communication rather than being required to correspond to individual objects or classes. Perceiver IO (Jaegle et al., 2022) encodes inputs into a latent array and decodes outputs through query-based cross-attention, while BiXT (Hiller et al., 2024) updates tokens and latents bidirectionally using a shared similarity matrix. RPAttention combines representatives constructed from the current input, explicit interaction among them, and a separately computed distribution map within an attention block that preserves the spatial token interface.

3 METHODOLOGY

3.1 OVERVIEW

Representative Attention (RPAttention) is an architectural replacement for attention modules in vision transformer backbones. It preserves the standard input-output interface and the surrounding spatial backbone, but replaces direct token-to-token interaction with a compact, content-adaptive token-space bottleneck.

Given an input feature map $\mathbf{X} \in \mathbb{R}^{B\times N\times C}$, where B is the batch size, N is the number of spatial tokens, and C is the channel dimension, RPAttention first computes the standard query, key, and value projections and reshapes them into h heads:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_q, \qquad \mathbf{K} = \mathbf{X}\mathbf{W}_k, \qquad \mathbf{V} = \mathbf{X}\mathbf{W}_v, \quad \mathbf{Q},\mathbf{K},\mathbf{V} \in \mathbb{R}^{B\times h\times N\times d}, \tag{1}$$

where $C = hd$. For reference, standard global self-attention computes

$$\mathrm{Attention}^S(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}, \tag{2}$$

which explicitly forms pairwise interactions among all N spatial tokens. RPAttention instead factors global communication through $M \ll N$ representatives:

$$\mathbf{O}^{\mathrm{RP}} = \underbrace{\mathrm{Distribute}\left(\mathbf{Q},\mathbf{K}_L,\mathrm{Interact}(\mathbf{V}_L)\right)}_{\text{spatial queries retrieve representative context}}, \qquad \{\mathbf{K}_L,\mathbf{V}_L\} = \underbrace{\mathrm{Gather}(\mathbf{K},\mathbf{V})}_{\text{tokens form representatives}}. \tag{3}$$

This formulation preserves the standard attention interface while constructing intermediate tokens through representation similarity rather than coordinate-based reduction. Figure 2 provides an overview of the complete Representative Attention module.

3.2 REPRESENTATION-CONDITIONED GATHERING

The gather step determines how spatial tokens are coarsened into a compact token-space bottleneck. Unlike fixed grid pooling or strided reduction, RPAttention constructs the representatives from learned

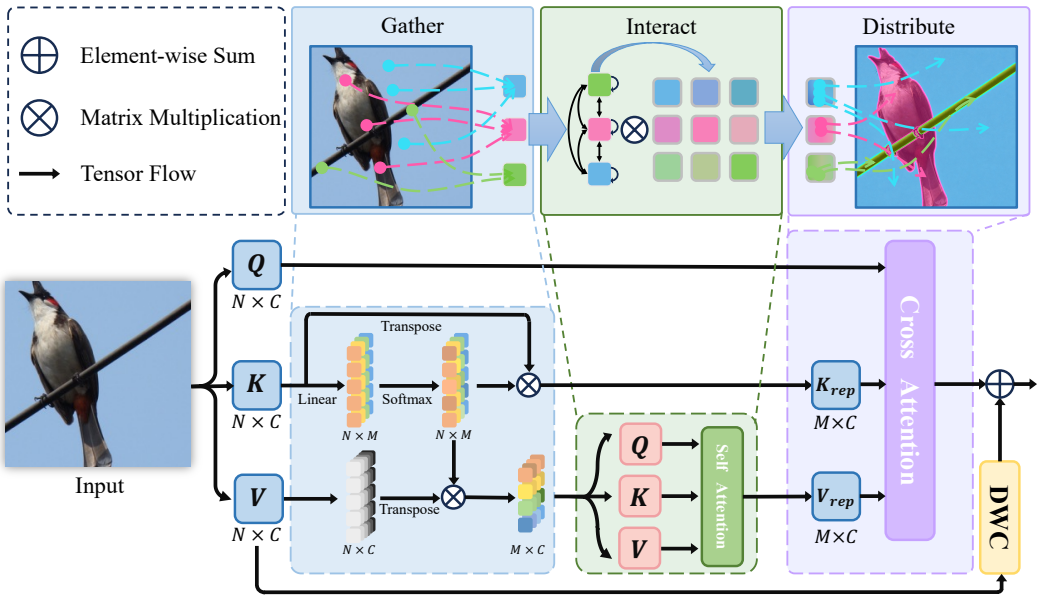

Figure 2: Overview of Representative Attention. Spatial tokens are gathered into a compact set of representatives according to representation similarity (*Gather*). These tokens then model global dependencies through lightweight self-attention in the compact token space (*Interact*). Finally, the refined representations are distributed back to all spatial tokens through query-driven cross-attention (*Distribute*). A parallel depth-wise convolution branch preserves local structural details.

feature compatibilities within each attention head. Let $\mathbf{W}_g \in \mathbb{R}^{d\times M}$ denote the gather projection, where $M \ll N$ is the number of representatives. The gather logits are computed from the key features as

$$\mathbf{A} = \mathrm{Softmax}_M(\mathbf{K}\mathbf{W}_g), \qquad \mathbf{A} \in \mathbb{R}^{B\times h\times N\times M}, \tag{4}$$

where the softmax is applied over the representative dimension M. For each attention head, the row $\mathbf{A}_{n,:}$ defines a distribution over the representatives for the n-th spatial token. Thus, $A_{n,m}$ specifies the fractional contribution of token n to representative m. It is determined by the learned feature compatibility between the token key and the corresponding representative anchor, namely the m-th column of $\mathbf{W}_g$. Consequently,

$$A_{n,m} \geq 0, \qquad \sum_{m=1}^{M} A_{n,m} = 1. \tag{5}$$

Each token can therefore contribute to multiple representatives, while the competition across the M representatives determines how its contribution is distributed. Because this routing depends on feature compatibility rather than token coordinates or predefined spatial regions, tokens at arbitrary spatial locations can contribute to the same representative. Different representatives may receive different total assignment masses. We normalize the incoming weights of each representative to keep the aggregated feature scales comparable:

$$\hat{\mathbf{A}}_{n,m} = \frac{\mathbf{A}_{n,m}}{\sum_{t=1}^{N} \mathbf{A}_{t,m} + \epsilon}. \tag{6}$$

In contrast to the row-normalized assignment matrix $\mathbf{A}$, the normalization of $\hat{\mathbf{A}}$ is performed over tokens for each representative. It converts the incoming assignment weights into representative-wise aggregation coefficients. This column-wise normalization makes the representative features less sensitive to the total assignment mass of each representative and keeps their scales comparable across the representative set. The normalized assignments are then used to construct the representative keys and values:

$$\mathbf{K}_L = \hat{\mathbf{A}}^\top \mathbf{K}, \qquad \mathbf{V}_L = \hat{\mathbf{A}}^\top \mathbf{V}, \qquad \mathbf{K}_L, \mathbf{V}_L \in \mathbb{R}^{B\times h\times M\times d}. \tag{7}$$

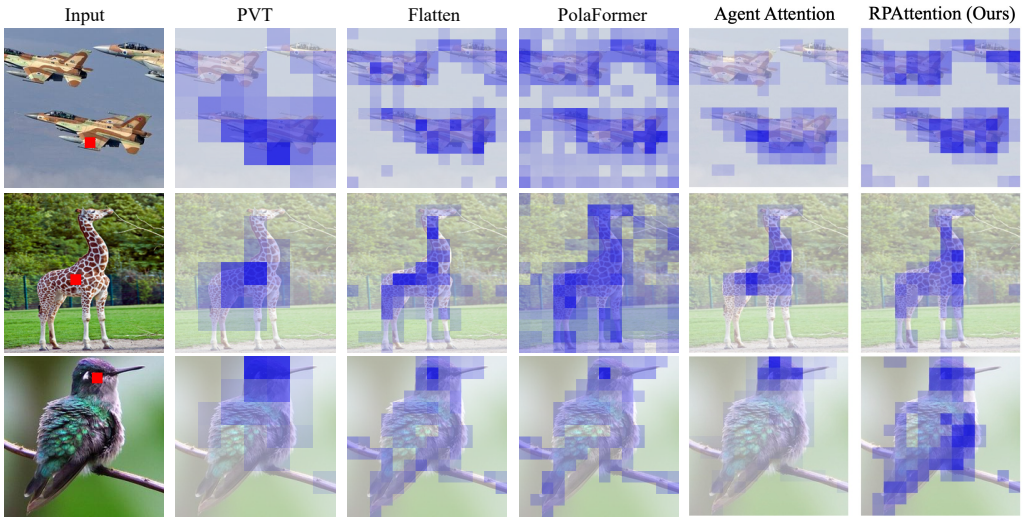

Figure 3: Qualitative comparison of query-to-space routing. Red squares mark query locations, and blue overlays show the corresponding response maps.

The representative keys $\mathbf{K}_L$ define the routing geometry used to address the compact latent space, whereas the representative values $\mathbf{V}_L$ carry the aggregated visual content that is later refined and distributed back to the spatial queries. This feed-forward construction produces input-derived representatives in a single pass. The representatives are therefore not fixed latent vectors: their values depend on the current token features and on the learned compatibility projection.

3.3 Latent Interaction

After gathering, RPAttention performs global interaction among the M representatives rather than among all N spatial tokens. We apply Layer Normalization to the representative keys and values, yielding $\bar{\mathbf{K}}_L = \mathrm{LN}_K(\mathbf{K}_L)$ and $\bar{\mathbf{V}}_L = \mathrm{LN}_V(\mathbf{V}_L)$. The latent interaction projections are derived from the representative values:

$$\widetilde{\mathbf{Q}}_L = \bar{\mathbf{V}}_L \mathbf{W}_q^L, \qquad \widetilde{\mathbf{K}}_L = \bar{\mathbf{V}}_L \mathbf{W}_k^L, \qquad \widetilde{\mathbf{V}}_L = \bar{\mathbf{V}}_L \mathbf{W}_v^L. \tag{8}$$

The refined representative content is then obtained by self-attention within the compact latent space:

$$\mathbf{Z}_L = \bar{\mathbf{V}}_L + \mathrm{Softmax}\left(\frac{\widetilde{\mathbf{Q}}_L \widetilde{\mathbf{K}}_L^\top}{\sqrt{d}}\right)\widetilde{\mathbf{V}}_L. \tag{9}$$

The interaction projections are derived from $\bar{\mathbf{V}}_L$, leaving $\bar{\mathbf{K}}_L$ unchanged for query-conditioned distribution. Because the interaction is performed among only M representatives, this stage has a quadratic cost in M rather than in the number of spatial tokens N.

3.4 Query-Conditioned Distribution

The refined representative content is returned to the spatial tokens through query-conditioned attention. Each spatial query attends to the normalized representative keys $\bar{\mathbf{K}}_L$:

$$\mathbf{O}_{\text{global}} = \mathrm{Softmax}\left(\frac{\mathbf{Q}\bar{\mathbf{K}}_L{}^\top}{\sqrt{d}}\right)\mathbf{Z}_L. \tag{10}$$

The distribution map is computed separately from, rather than taken directly from, the gathering assignments. Thus, a token's contribution to a representative need not determine how it retrieves the refined content. The compact global bottleneck may weaken local boundaries and other fine-grained details. To preserve this information, we introduce a parallel local bypass implemented with a

depth-wise convolution (DWC), following FLatten Transformer (Han et al., 2023). The global and local paths are fused before the output projection:

$$\mathbf{X}_{\text{out}} = (\text{MergeHeads}(\mathbf{O}_{\text{global}}) + \text{DWC}(\mathbf{X}\mathbf{W}_v))\,\mathbf{W}_o. \tag{11}$$

Here, MergeHeads concatenates the h attention heads into C channels. The representative path captures global context, while the DWC path preserves local spatial details. Their fusion combines long-range interaction with local structure preservation without changing the spatial input–output interface.

3.5 Complexity Analysis

RPAttention can replace an attention module without changing the surrounding transformer layer, patch embedding, MLP, or residual structure. Let N denote the number of spatial tokens, $C = hd$ the channel dimension, and M the number of representatives. Ignoring lower-order normalization and dropout terms, the computational cost of one RPAttention block is

$$\Omega_{\text{RP}} = \underbrace{4NC^2}_{\text{Projections}} + \underbrace{3NMC}_{\text{Gather}} + \underbrace{3MCd + 2M^2C}_{\text{Interaction}} + \underbrace{2NMC}_{\text{Distribute}} + \underbrace{k^2NC}_{\text{DWC}}, \tag{12}$$

where $C = hd$ is the channel dimension and $k = 3$ is the kernel size of the depth-wise convolution.

The first term accounts for the three input projections and the output projection. In the gathering stage, the projection $\mathbf{K}\mathbf{W}_g$ costs NMC, while aggregating the keys and values into representatives costs an additional $2NMC$. The latent interaction requires $3MCd$ for the head-wise latent QKV projections and $2M^2C$ for the representative-level attention matrix and value aggregation. The distribution stage contributes another $2NMC$ for computing query-to- representative affinities and reading out the refined representative content. For comparison, standard global self-attention has cost

$$\Omega_{\text{SA}} = 4NC^2 + 2N^2C, \tag{13}$$

where the second term accounts for the $N \times N$ attention affinities and the corresponding value aggregation. Thus, RPAttention replaces the quadratic token-interaction term $2N^2C$ with

$$5NMC + 3MCd + 2M^2C + k^2NC. \tag{14}$$

When M, C, and k are fixed with respect to N, the token-dependent complexity of RPAttention scales linearly with the number of spatial tokens:

$$\Omega_{\text{RP}} = \mathcal{O}\big(NC^2 + NMC + MCd + M^2C + k^2NC\big) = \mathcal{O}(N). \tag{15}$$

Unlike local or windowed attention, this linear scaling does not require restricting queries to spatial neighborhoods. Every spatial query can access the same compact set of globally gathered representatives, preserving a global communication path.

4 Experimental Validation

4.1 Experimental Setup

We organize our experiments into three categories: image classification, dense prediction, and practical efficiency. ImageNet-1K (Deng et al., 2009) is used for classification, COCO (Lin et al., 2014) for object detection and instance segmentation, and ADE20K (Zhou et al., 2019) for semantic segmentation. We further measure inference latency and throughput at different input resolutions to evaluate the practical scaling behavior of RPAttention.

We instantiate RPAttention as a replacement for the attention blocks in four backbone families: DeiT (Touvron et al., 2021), PVT (Wang et al., 2021), PVTv2 (Wang et al., 2022), and Swin Transformer (Liu et al., 2021). All other backbone components and task-specific prediction heads are kept unchanged. The main tables compare each modified backbone with its original attention-based counterpart. Detailed training settings are provided in Appendix B, and measurement protocols and additional practical-efficiency comparisons in Appendix C.

Table 1: ImageNet-1K classification results across four backbone families.

Method	Params	FLOPs	Top-1
DeiT-T	5.7M	1.2G	72.2
+ RPAttention	5.9M	1.2G	**75.4 (+3.2)**
DeiT-S	22.1M	4.6G	79.8
+ RPAttention	22.3M	4.5G	**80.8 (+1.0)**
PVT-T	13.2M	1.9G	75.1
+ RPAttention	11.5M	2.2G	**79.2 (+4.1)**
PVT-S	24.5M	3.8G	79.8
+ RPAttention	20.5M	4.4G	**82.5 (+2.7)**
PVT-M	44.2M	6.7G	81.2
+ RPAttention	35.5M	7.5G	**83.5 (+2.3)**

Method	Params	FLOPs	Top-1
PVTv2-B0	3.7M	0.5G	70.5
+ RPAttention	3.2M	0.7G	**73.1 (+2.6)**
PVTv2-B1	13.1M	2.1G	78.7
+ RPAttention	12.3M	2.4G	**80.5 (+1.8)**
PVTv2-B2	25.4M	4.0G	82.0
+ RPAttention	21.3M	4.6G	**82.6 (+0.6)**
Swin-T	29.0M	4.5G	81.3
+ RPAttention	28.4M	4.9G	**82.7 (+1.4)**
Swin-S	50M	8.7G	83.0
+ RPAttention	50M	9.3G	**83.8 (+0.8)**

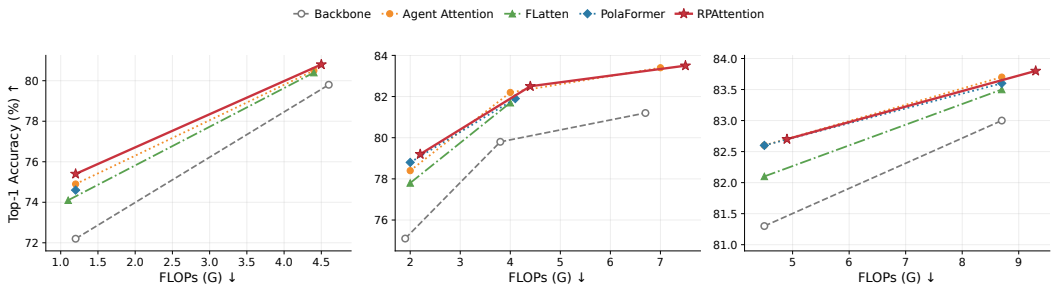

Figure 4: ImageNet-1K accuracy–FLOPs comparison with efficient-attention methods. From left to right: DeiT, PVT, and Swin. Points along each curve follow increasing model size.

4.2 ImageNet-1K Classification

Table 1 reports ImageNet-1K classification results across four backbone families and ten model configurations. RPAttention consistently improves Top-1 accuracy over the corresponding backbone across all evaluated configurations, with gains ranging from $+0.6$ to $+4.1$ percentage points. For example, RPAttention improves PVT-T/PVT-S/PVT-M from $75.1/79.8/81.2$ to $79.2/82.5/83.5$, respectively, and improves Swin-S from 83.0 to 83.8. These results demonstrate that the proposed method is effective across both flat and hierarchical vision transformer backbones. Figure 4 further compares RPAttention with other efficient-attention methods on DeiT, PVT, and Swin. Detailed numerical comparisons are provided in Appendix D.1.

Figure 3 provides a qualitative comparison of query-to-space routing. RPAttention responds to spatially separated aircraft regions and extends the bird-head query to the body, whereas PVT and Agent Attention show more localized responses in the bird example. In the aircraft and giraffe examples, RPAttention also exhibits less background spread than PolaFormer. These examples illustrate how the representative bottleneck can support broad, content-dependent responses while retaining object-related spatial structure. Additional visualizations are provided in Appendix E.3.

4.3 Dense Prediction

We evaluate RPAttention on COCO 2017 (Lin et al., 2014) for object detection and instance segmentation using Mask R-CNN (He et al., 2017) with the $1\times$ schedule, and on ADE20K (Zhou et al., 2019) for semantic segmentation. Table 2 compares RPAttention with the corresponding backbone baselines across the evaluated configurations. Additional comparisons on COCO and ADE20K are provided in Appendix D.2.

Object detection and instance segmentation. RPAttention improves box AP across all evaluated configurations by 0.5–3.9 points and generally improves mask AP. The largest gains are $+3.9$ box

Table 2: Dense prediction results on COCO and ADE20K. FLOPs are computed at 1280×800 for COCO and 512×2048 for ADE20K. Unavailable entries are denoted by "–".

Method	Detection & Instance Seg.							Semantic Seg.			
	FLOPs	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}	FLOPs	Params	mIoU	mAcc
PVT-T	240G	36.7	59.2	39.3	35.1	56.7	37.3	158G	17M	36.57	46.72
+ RPAttention	215G	40.4	62.8	43.8	37.7	59.9	40.4	128G	15M	40.00	53.54
PVT-S	305G	40.4	62.9	43.8	37.8	60.1	40.3	225G	28M	41.95	53.02
+ RPAttention	259G	44.1	66.6	48.5	40.8	63.7	44.3	177G	24M	43.25	56.04
PVT-M	392G	42.0	64.4	45.6	39.0	61.6	42.1	315G	48M	42.91	53.80
+ RPAttention	322G	45.9	68.2	50.4	41.9	65.4	45.3	241G	39M	45.16	58.13
Swin-T	267G	43.7	66.6	47.7	39.8	63.3	42.7	945G	60M	44.51	55.61
+ RPAttention	271G	44.5	67.3	48.4	40.5	64.2	43.3	949G	60M	46.97	59.17
Swin-S	358G	45.7	67.9	50.4	41.1	64.9	44.2	1038G	81M	47.64	58.78
+ RPAttention	360G	46.3	69.0	50.4	41.8	65.7	45.1	1042G	81M	48.48	60.12
PVTv2-B0	196G	38.2	60.5	40.7	36.2	57.8	38.6	113G	7.6M	37.2	–
+ RPAttention	184G	38.7	61.0	41.4	36.1	57.9	38.5	100G	7.1M	37.93	49.08
PVTv2-B1	244G	41.8	64.3	45.9	38.8	61.2	41.6	162G	17.8M	42.5	–
+ RPAttention	220G	43.2	65.6	47.3	39.8	62.5	42.7	136G	16.0M	43.10	54.86
PVTv2-B2	309G	45.3	67.1	49.6	41.2	64.2	44.4	230G	29.1M	45.2	–
+ RPAttention	264G	46.2	68.1	50.6	42.0	64.9	45.3	182G	25.1M	46.69	58.54

Table 3: Component ablations on PVTv2-B0.

Variant	Params	FLOPs	Top-1
PVTv2-B0	3.7M	0.5G	70.5
w/o DWC	3.24M	0.71G	72.7 (+2.2)
w/ k-means routing	3.29M	0.78G	72.6 (+2.1)
w/o Interaction	3.23M	0.70G	72.8 (+2.3)
RPAttention	3.25M	0.72G	**73.1 (+2.6)**

Table 4: Representative counts on PVTv2-B0.

Reps. per stage	Top-1	mIoU	mAcc
$(9, 16, 25, 25)$	72.8	–	–
$(9, 16, 49, 49)$	**73.1**	37.21	48.72
$(49, 49, 49, 49)$	72.9	–	–
$(96, 64, 64, 49)$	73.0	–	–
$(96, 64, 49, 49)$	**73.1**	**37.93**	**49.08**

AP on PVT-M and $+3.0$ mask AP on PVT-S, while PVTv2-B0 shows a minor 0.1 mask AP decrease. PVT and PVTv2 variants also use fewer FLOPs than their corresponding backbones.

Semantic segmentation. RPAttention improves mIoU across all eight configurations by 0.60–3.43 points. For PVT and PVTv2, these gains are accompanied by fewer parameters and lower FLOPs, while Swin improves mIoU at a similar parameter count and computational cost.

Efficiency scaling. We further examine practical efficiency under increasing spatial workloads. As input resolution grows, RPAttention-PVT-T becomes faster than PVT-T and the compared efficient-attention variants, while using less peak memory than PVT-T (Figure 5, top). Its peak memory is comparable to Agent Attention. A complementary study varies the attention-window size of Swin-T while keeping RPAttention-Swin-T fixed, showing increasingly favorable FLOPs and throughput comparisons as the window size grows (Figure 5, bottom).

4.4 Ablation Studies

Component analysis. Table 3 examines the contributions of local refinement, representative interaction, and learned gathering on PVTv2-B0, with each variant compared against the full model. Removing the DWC branch reduces Top-1 accuracy from 73.1% to 72.7%, while retaining a 2.2-point gain over the original backbone. Thus, the improvement does not rely solely on the local convolu-

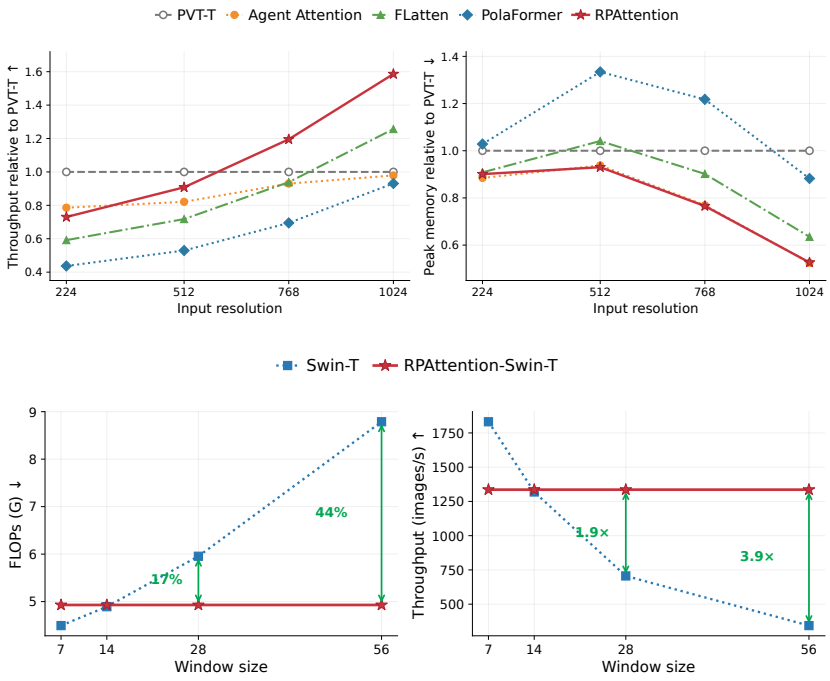

Figure 5: Practical efficiency on an RTX 3090. Top: matched-batch throughput and batch-1 peak GPU memory across input resolutions, normalized to PVT-T. Bottom: FLOPs and throughput as the Swin-T attention window grows, with RPAttention-Swin-T held fixed.

tional branch, although local refinement provides an additional benefit. Removing representative interaction reduces accuracy to 72.8%, suggesting that exchanging information among representatives is beneficial beyond gathering and distribution alone. We replace the learned projection in Gather with per-input k-means routing using three iterations, leaving the remaining components unchanged. This variant achieves 72.6% accuracy at 0.78G FLOPs, whereas learned gathering achieves 73.1% at 0.72G, providing higher accuracy at lower computational cost.

Number of representatives. Table 4 compares five stage-wise representative configurations on PVTv2-B0. ImageNet-1K Top-1 accuracy varies only from 72.8% to 73.1%. Early stages operate at higher spatial resolutions, where a small representative budget may overly compress spatial detail needed for dense prediction. With the last two stages fixed at $(49, 49)$, increasing the early-stage counts from $(9, 16)$ to $(96, 64)$ maintains 73.1% Top-1 accuracy while improving ADE20K mIoU by 0.72 points and mAcc by 0.36 points. These results motivate our default choice of $(96, 64, 49, 49)$.

Additional analyses of cross-seed routing stability, foreground alignment, and query-response visualizations are provided in Appendices E.1, E.2, and E.3, respectively.

5 Conclusion

We introduced Representative Attention (RPAttention), an efficient global attention module that constructs compact representatives from input features rather than fixed spatial partitions. Its *Gather–Interact–Distribute* pipeline gathers tokens through learned feature compatibility, enables interaction among representatives, and distributes the refined information according to each query, while preserving the spatial token interface. With M representatives and N spatial tokens, the cost of token interactions is $\mathcal{O}(NM + M^2)$. Experiments on ImageNet-1K classification, COCO object detection and instance segmentation, and ADE20K semantic segmentation demonstrate improved overall performance over the corresponding backbone baselines and competitive results against recent efficient attention methods. These findings support learning token compression from visual features as an effective alternative to spatially defined proxy construction for global visual communication.

Reproducibility Statement

Appendix A.1 provides a PyTorch-style implementation of the core Gather–Interact–Distribute module. Appendix A.2 lists the stage-wise architectures and representative counts of all evaluated models. Appendix B documents the training and evaluation recipes for ImageNet-1K, COCO, and ADE20K, including optimization and augmentation settings. For efficiency measurements, Appendix C details the runtime environment and measurement protocol. Upon acceptance, we will release the complete training and evaluation code, configurations, scripts, and principal checkpoints.

AI Use Statement

AI tools were used to polish the language and improve readability. The authors thoroughly reviewed the revised text and take full responsibility for the final manuscript.

References

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers. In *ICLR*, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

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 *CVPR*, 2022.

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

Qihang Fan, Yuang Ai, Huaibo Huang, and Ran He. Random wins all: Rethinking grouping strategies for vision tokens. In *CVPR*, 2026.

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

Dongchen Han, Yifan Pu, Zhuofan Xia, Yizeng Han, Xuran Pan, Xiu Li, Jiwen Lu, Shiji Song, and Gao Huang. Bridging the divide: Reconsidering softmax and linear attention. In *NeurIPS*, 2024a.

Dongchen Han, Tianzhu Ye, Yizeng Han, Zhuofan Xia, Siyuan Pan, Pengfei Wan, Shiji Song, and Gao Huang. Agent attention: On the integration of softmax and linear attention. In *ECCV*, 2024b.

Dongchen Han, Tianyu Li, Ziyi Wang, and Gao Huang. Vision transformers are circulant attention learners. In *AAAI*, 2026.

Ali Hassani, Steven Walton, Jiachen Li, Shen Li, and Humphrey Shi. Neighborhood attention transformer. In *CVPR*, 2023.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

Markus Hiller, Krista A. Ehinger, and Tom Drummond. Perceiving longer sequences with bi-directional cross-attention transformers. In *NeurIPS*, 2024.

Huaibo Huang, Xiaoqiang Zhou, Jie Cao, Ran He, and Tieniu Tan. Vision transformer with super token sampling. In *CVPR*, 2023.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, et al. Perceiver IO: A general architecture for structured inputs & outputs. In *ICLR*, 2022.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *ICML*, 2020.

Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *CVPR*, 2019.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *NeurIPS*, 2020.

Weikang Meng, Yadan Luo, Xin Li, Dongmei Jiang, and Zheng Zhang. PolaFormer: Polarity-aware linear attention for vision transformers. In *ICLR*, 2025.

Michael Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. Token-learner: Adaptive space-time tokenization for videos. In *NeurIPS*, 2021.

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

Sinong Wang, Belinda Z Li, Madian Khabsa, Hanwen Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

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. In *ICCV*, 2021.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. PVT v2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, 2022.

Zhuofan Xia, Xuran Pan, Shiji Song, Li Erran Li, and Gao Huang. Vision transformer with deformable attention. In *CVPR*, 2022.

Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A Nyström-based algorithm for approximating self-attention. In *AAAI*, 2021.

Jiarui Xu, Shalini De Mello, Sifei Liu, Wonmin Byeon, Thomas Breuel, Jan Kautz, and Xiaolong Wang. GroupViT: Semantic segmentation emerges from text supervision. In *CVPR*, 2022.

Zhiyu Yao, Jian Wang, Haixu Wu, Jingdong Wang, and Mingsheng Long. Mobile attention: mobile-friendly linear-attention for vision transformers. In *ICML*, 2024.

Qihang Yu, Huiyu Wang, Siyuan Qiao, Maxwell Collins, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. k-means mask transformer. In *ECCV*, 2022.

Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ADE20K dataset. *IJCV*, 2019.

Lei Zhu, Xinjiang Wang, Zhanghan Ke, Wayne Zhang, and Rynson W. H. Lau. BiFormer: Vision transformer with bi-level routing attention. In *CVPR*, 2023.

Appendix

A Implementation and Model Architectures

A.1 Implementation of RPAttention

We provide a compact PyTorch-style implementation of the core *Gather–Interact–Distribute* computation below, omitting standard module initialization for clarity.

```
def rpattention_forward(x, H, W):
    B, N, C = x.shape
    qkv = self.qkv(x).reshape(B, N, 3, self.num_heads, self.
   head_dim).permute(2, 0, 3, 1, 4)
    q, k, v = qkv.unbind(0)

    # Local depth-wise convolution branch
    v_img = v.transpose(1, 2).reshape(B, N, C)
    v_img = v_img.transpose(1, 2).reshape(B, C, H, W)
    local = self.dwc(v_img).flatten(2).transpose(1, 2)

    # Gather: tokens -> representatives
    A = F.softmax(self.gather_proj(k), dim=-1)
    A = A / A.sum(dim=2, keepdim=True).clamp_min(1e-6)
    gather = A.transpose(-1, -2)

    K_l = self.latent_k_norm(gather @ k)
    V_l = self.latent_v_norm(gather @ v)

    # Interact: self-attention among representatives
    lqkv = self.latent_qkv(V_l).reshape(
        B, self.num_heads, self.num_latents, 3, self.head_dim
    ).permute(3, 0, 1, 2, 4)
    lq, lk, lv = lqkv.unbind(0)

    attn_l = (lq @ lk.transpose(-2, -1)) * self.scale
    V_l = V_l + F.softmax(attn_l, dim=-1) @ lv

    # Distribute: representatives -> tokens
    attn = (q @ K_l.transpose(-2, -1)) * self.scale
    global_out = F.softmax(attn, dim=-1) @ V_l
    global_out = global_out.transpose(1, 2).reshape(B, N, C)

    return self.out_proj(global_out + local)
```

A.2 Model Architectures

We summarize the architectures of the RPAttention models used in the main experiments. Spatial resolutions are reported for a 224×224 input. For each stage, we list the embedding dimension, number of attention heads, number of representatives per head, and block depth. The notation $\times L$ denotes L repeated blocks, and LN denotes LayerNorm. RPAttention replaces the attention module in every block, while the remaining backbone components are preserved.

Table A.1: Architectures of RPAttention-DeiT models.

Stage	Resolution	RPAttention-DeiT-T	RPAttention-DeiT-S
Patch embedding		Patch 16×16, stride 16	
res1	14×14	$\begin{bmatrix} \text{dim } 192 \\ \text{head } 3 \\ \text{rep. } 49 \end{bmatrix} \times 12$	$\begin{bmatrix} \text{dim } 384 \\ \text{head } 6 \\ \text{rep. } 49 \end{bmatrix} \times 12$

Table A.2: Architectures of RPAttention-PVT models.

Stage	Resolution	RPAttention-PVT-T	RPAttention-PVT-S	RPAttention-PVT-M
Patch embedding		Conv 4×4, stride 4, 64, LN		
res1	56×56	$\begin{bmatrix} \text{dim } 64 \\ \text{head } 1 \\ \text{rep. } 96 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{dim } 64 \\ \text{head } 1 \\ \text{rep. } 96 \end{bmatrix} \times 3$	$\begin{bmatrix} \text{dim } 64 \\ \text{head } 1 \\ \text{rep. } 96 \end{bmatrix} \times 3$
Patch embedding		Conv 2×2, stride 2, 128, LN		
res2	28×28	$\begin{bmatrix} \text{dim } 128 \\ \text{head } 2 \\ \text{rep. } 64 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{dim } 128 \\ \text{head } 2 \\ \text{rep. } 64 \end{bmatrix} \times 4$	$\begin{bmatrix} \text{dim } 128 \\ \text{head } 2 \\ \text{rep. } 64 \end{bmatrix} \times 4$
Patch embedding		Conv 2×2, stride 2, 320, LN		
res3	14×14	$\begin{bmatrix} \text{dim } 320 \\ \text{head } 5 \\ \text{rep. } 49 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{dim } 320 \\ \text{head } 5 \\ \text{rep. } 49 \end{bmatrix} \times 6$	$\begin{bmatrix} \text{dim } 320 \\ \text{head } 5 \\ \text{rep. } 49 \end{bmatrix} \times 18$
Patch embedding		Conv 2×2, stride 2, 512, LN		
res4	7×7	$\begin{bmatrix} \text{dim } 512 \\ \text{head } 8 \\ \text{rep. } 49 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{dim } 512 \\ \text{head } 8 \\ \text{rep. } 49 \end{bmatrix} \times 3$	$\begin{bmatrix} \text{dim } 512 \\ \text{head } 8 \\ \text{rep. } 49 \end{bmatrix} \times 3$

Table A.3: Architectures of RPAttention-PVTv2 models.

Stage	Resolution	RPAttention-PVTv2-B0	RPAttention-PVTv2-B1	RPAttention-PVTv2-B2
res1	56×56	$\begin{bmatrix} \text{dim } 32 \\ \text{head } 1 \\ \text{rep. } 96 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{dim } 64 \\ \text{head } 1 \\ \text{rep. } 96 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{dim } 64 \\ \text{head } 1 \\ \text{rep. } 96 \end{bmatrix} \times 3$
res2	28×28	$\begin{bmatrix} \text{dim } 64 \\ \text{head } 2 \\ \text{rep. } 64 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{dim } 128 \\ \text{head } 2 \\ \text{rep. } 64 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{dim } 128 \\ \text{head } 2 \\ \text{rep. } 64 \end{bmatrix} \times 4$
res3	14×14	$\begin{bmatrix} \text{dim } 160 \\ \text{head } 5 \\ \text{rep. } 49 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{dim } 320 \\ \text{head } 5 \\ \text{rep. } 49 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{dim } 320 \\ \text{head } 5 \\ \text{rep. } 49 \end{bmatrix} \times 6$
res4	7×7	$\begin{bmatrix} \text{dim } 256 \\ \text{head } 8 \\ \text{rep. } 49 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{dim } 512 \\ \text{head } 8 \\ \text{rep. } 49 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{dim } 512 \\ \text{head } 8 \\ \text{rep. } 49 \end{bmatrix} \times 3$

Table A.4: Architectures of RPAttention-Swin models.

Stage	Resolution	RPAttention-Swin-T	RPAttention-Swin-S
Patch embedding		Patch 4×4, stride 4, 96, LN	
res1	56×56	$\begin{bmatrix} \text{dim } 96 \\ \text{head } 3 \\ \text{rep. } 96 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{dim } 96 \\ \text{head } 3 \\ \text{rep. } 96 \end{bmatrix} \times 2$
Patch merging		2×2, 192, LN	
res2	28×28	$\begin{bmatrix} \text{dim } 192 \\ \text{head } 6 \\ \text{rep. } 64 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{dim } 192 \\ \text{head } 6 \\ \text{rep. } 64 \end{bmatrix} \times 2$
Patch merging		2×2, 384, LN	
res3	14×14	$\begin{bmatrix} \text{dim } 384 \\ \text{head } 12 \\ \text{rep. } 49 \end{bmatrix} \times 6$	$\begin{bmatrix} \text{dim } 384 \\ \text{head } 12 \\ \text{rep. } 49 \end{bmatrix} \times 18$
Patch merging		2×2, 768, LN	
res4	7×7	$\begin{bmatrix} \text{dim } 768 \\ \text{head } 24 \\ \text{rep. } 49 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{dim } 768 \\ \text{head } 24 \\ \text{rep. } 49 \end{bmatrix} \times 2$

B Training Details

B.1 ImageNet-1K Classification

Table B.1: Training configurations for ImageNet-1K classification.

Setting	Value
Input resolution	224×224
Training length	300 epochs
Optimizer	AdamW
Weight decay	0.05
LR schedule	Cosine decay to 10^{-5}
Warm-up	20 epochs, linear from 10^{-6}
RandAugment	rand-m9-mstd0.5-inc1
Mixup / CutMix	0.8 / 1.0
Label smoothing	0.1
Random erasing	0.25
Random resized crop	scale $(0.08, 1.0)$, ratio $(0.75, 1.33)$
Horizontal flip	0.5
Drop-path rate	Model-specific (see below)
EMA	No
Default seed	42
Hardware	TPU v4-32
Local batch size	256 per process
Global batch size	4096
Peak learning rate	4×10^{-3}

All RPAttention classification models reported in the main tables are trained on a TPU v4-32 with a global batch size of 4096 and a peak learning rate of 4×10^{-3}. The drop-path rates are 0.0/0.1 for DeiT-T/S, 0.0/0.1/0.3 for PVT-T/S/M, 0.0/0.1/0.2 for PVTv2-B0/B1/B2, and 0.2/0.4 for Swin-T/S. The remaining settings in Table B.1 are shared across these models.

B.2 Dense Prediction

Table B.2: Training configurations for COCO object detection and instance segmentation.

Setting	PVT / PVTv2	Swin
Framework	Mask R-CNN	
Training length	$1\times$ (12 epochs)	
Optimizer	AdamW	
Weight decay	1×10^{-4}	5×10^{-2}
Global batch size	16	
Learning rate	2×10^{-4}	1×10^{-4}
LR schedule	Step decay at epochs 8 and 11	
Warm-up	500 iterations, linear, ratio 10^{-3}	
Test resize scale ($W \times H$)	1333×800, keep ratio	
Test augmentation	Single scale, no flipping	

For PVT and PVTv2, COCO training uses a resize scale of $(1333, 800)$ with aspect ratio preserved. For Swin, training uses multiscale resizing with short-side lengths from 480 to 800 in steps of 32 and a long-side limit of 1333, with an additional resize–crop–resize augmentation branch. Both settings use random horizontal flipping with probability 0.5.

Table B.3: Training configurations for ADE20K semantic segmentation.

Setting	PVT / PVTv2	Swin
Segmentation head	Semantic FPN (Kirillov et al., 2019)	UPerNet (Xiao et al., 2018)
Optimizer	AdamW	
Global batch size	16 / 32	16
Training crop	512×512	
Resize scale range	$[0.5, 2.0]$	
Horizontal flip probability	0.5	
Photometric distortion	Yes	
Test resize scale ($W \times H$)	2048×512, keep ratio	
Test augmentation	Single scale, no flipping	
Training iterations	40,000	160,000
Learning rate	2×10^{-4}	6×10^{-5}
Weight decay	1×10^{-4}	1×10^{-2}
LR schedule	Poly ($p = 0.9$)	Poly ($p = 1.0$)
Minimum learning rate	0	
Warm-up	None	1,500 iterations, linear

All RPAttention backbones used for downstream tasks are initialized from their corresponding ImageNet-1K checkpoints. Tables B.2 and B.3 provide reference training configurations for COCO and ADE20K, respectively.

C Practical Efficiency

C.1 Measurement protocol

All practical-efficiency measurements in Figure 5 and this section are conducted on a single otherwise idle NVIDIA RTX 3090 using eager PyTorch 2.7.1, CUDA 11.8, cuDNN 9.1, and timm 0.4.12. We use AMP/FP16 inference with TF32 disabled, `model.eval()`, and `torch.inference_mode()`, without `torch.compile` or inference-specific operator fusion. CPU intra-operation parallelism is restricted to one thread, while cuDNN benchmarking is enabled.

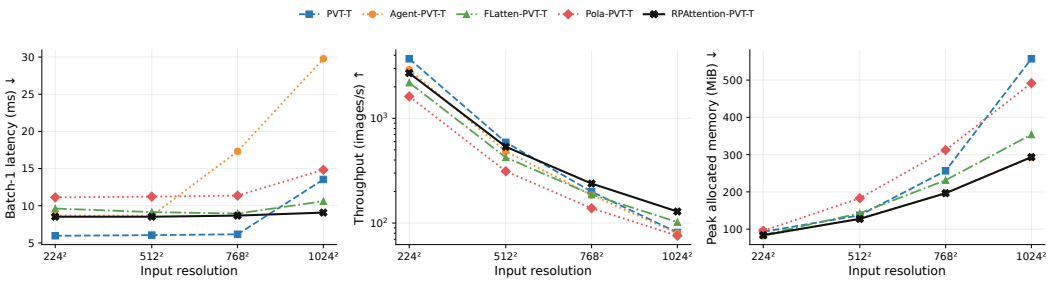

Figure C.1: Absolute execution-cost measurements for PVT-T and representative efficient-attention variants on a single NVIDIA RTX 3090. Left: synchronized batch-1 model-forward latency. Middle: matched-batch throughput, using batch sizes 64, 16, 8, and 4 at 224^2, 512^2, 768^2, and 1024^2, respectively. Right: batch-1 peak allocated GPU memory. All methods use eager AMP/FP16 inference under the same software and measurement protocol.

Each model–resolution–batch configuration is evaluated in a fresh process. Input tensors are generated in advance and remain resident on the GPU; data loading, preprocessing, host-to-device transfer, and task-specific post-processing are excluded. The models are randomly initialized for execution-cost measurement. This does not alter the measured execution graph because all compared methods use dense computation without input-dependent token pruning or conditional execution.

Latency is measured as synchronized wall-clock model-forward latency with batch size 1. Throughput uses batch sizes 64, 16, 8, and 4 at resolutions 224^2, 512^2, 768^2, and 1024^2, respectively, with the batch size fixed across methods at each resolution. Peak allocated memory is measured after warm-up and after resetting the PyTorch peak-memory counter. Figures report memory at batch size 1; Table C.1 additionally reports memory at the batch size used for throughput measurement. Each memory value is measured once per configuration. Each runtime measurement consists of five rounds, each with 50 warm-up forwards followed by 200 timed forwards. We report the mean across the five round means; Table C.1 also reports their standard deviation. The maximum coefficient of variation among the throughput measurements in Figure 5 is below 0.36%.

For PVT-based models, each network is instantiated at its target resolution so that its absolute positional embeddings match the input resolution; all methods follow the same construction rule. Therefore, the high-resolution sweep measures execution-cost scaling rather than high-resolution classification accuracy. PVT, Agent Attention, FLatten, and PolaFormer use their official implementations.

C.2 Absolute Execution Cost

Figure C.1 presents absolute execution costs across input resolutions, complementing the normalized results in the top panels of Figure 5. Table C.1 provides the corresponding numerical results and runtime variability, together with peak memory at the throughput batch size. At 224^2, RPAttention-PVT-T incurs additional execution overhead relative to PVT-T, with a batch-1 latency of 8.52 ms versus 5.96 ms and a throughput of 2706 versus 3706 images/s. At 768^2, RPAttention achieves higher throughput than PVT-T, although its batch-1 latency remains higher. At 1024^2, RPAttention reduces batch-1 latency from 13.53 ms to 9.08 ms and increases matched-batch throughput from 81.3 to 129.0 images/s. Its batch-1 peak allocated memory decreases from 557.2 MiB to 293.3 MiB.

At 1024^2, RPAttention provides $1.62\times$, $1.26\times$, and $1.70\times$ the throughput of Agent Attention, FLatten, and PolaFormer, respectively. Its peak memory is comparable to Agent Attention, while remaining lower than FLatten and PolaFormer. These measurements demonstrate a resolution-dependent crossover rather than a universal speed advantage at every operating point. The observed trend is consistent with the fixed-representative interaction cost $\mathcal{O}(NM + M^2)$ becoming more favorable as the spatial-token count N increases.

Table C.1: Absolute inference cost of PVT-T and its attention variants on an RTX 3090 under the protocol above. Latency uses batch size 1; throughput uses batch size B. Runtime values report mean $\pm$ standard deviation across five rounds. M_1 and M_B denote peak allocated memory at batch sizes 1 and B, respectively, each measured once after warm-up.

Input	**Method**	B	**Latency**$\downarrow$ (ms)	**Throughput**$\uparrow$ (images/s)	$M_1 \downarrow$ (MiB)	$M_B \downarrow$ (MiB)
224^2	PVT-T	64	5.961 ± 0.015	3706.11 ± 6.36	93.0	662.0
	Agent Attention	64	8.690 ± 0.032	2912.82 ± 6.53	82.3	652.0
	FLatten	64	9.625 ± 0.053	2192.96 ± 2.02	84.5	654.2
	PolaFormer	64	11.127 ± 0.008	1619.20 ± 0.95	95.6	1046.7
	RPAttention	64	8.523 ± 0.046	2706.24 ± 1.57	83.8	652.7
512^2	PVT-T	16	6.044 ± 0.019	590.28 ± 0.63	137.1	841.5
	Agent Attention	16	8.655 ± 0.031	484.63 ± 0.96	128.6	832.6
	FLatten	16	9.144 ± 0.041	423.61 ± 0.60	142.8	848.5
	PolaFormer	16	11.224 ± 0.038	312.53 ± 0.18	183.0	1350.1
	RPAttention	16	8.522 ± 0.032	535.90 ± 0.28	127.6	832.7
768^2	PVT-T	8	6.158 ± 0.011	199.49 ± 0.22	256.5	1345.1
	Agent Attention	8	17.319 ± 0.008	185.51 ± 0.13	197.4	938.9
	FLatten	8	8.946 ± 0.038	187.24 ± 0.09	231.3	971.0
	PolaFormer	8	11.365 ± 0.045	138.56 ± 0.09	312.4	1536.1
	RPAttention	8	8.672 ± 0.062	238.42 ± 0.15	196.4	938.5
1024^2	PVT-T	4	13.535 ± 0.004	81.30 ± 0.04	557.2	1891.3
	Agent Attention	4	29.775 ± 0.004	79.58 ± 0.02	292.6	856.5
	FLatten	4	10.608 ± 0.021	102.16 ± 0.06	353.9	917.3
	PolaFormer	4	14.834 ± 0.004	75.67 ± 0.05	491.7	1418.7
	RPAttention	4	9.077 ± 0.021	128.95 ± 0.07	293.3	855.9

Table C.2: Relative changes versus the corresponding PVT backbone. Higher throughput and lower latency or memory are better.

	Throughput change				Change at 1024^2	
Scale	224^2	512^2	768^2	1024^2	Latency	Memory
Tiny	-27.0%	-9.2%	$+19.5\%$	$+58.6\%$	-32.9%	-47.4%
Small	-27.7%	-8.1%	$+21.9\%$	$+62.8\%$	-33.6%	-44.5%
Medium	-28.3%	-6.2%	$+24.7\%$	$+65.4\%$	-24.0%	-41.1%

C.3 Scaling across model sizes

Table C.2 extends the resolution sweep from PVT-Tiny to PVT-Small and PVT-Medium. All three model sizes exhibit the same matched-batch throughput crossover: RPAttention is slower at 224^2 and 512^2, becomes faster at 768^2, and provides a 58.6%–65.4% throughput improvement at 1024^2. Batch-1 latency and peak memory also decrease consistently at 1024^2. The observed crossover is therefore not specific to the Tiny model.

C.4 Interaction-range scaling

Figure 5 (bottom) varies Swin-T's window size while fixing RPAttention-Swin-T to RRRR with window size 56 and $(96, 64, 49, 49)$ representatives. All models use 224×224 inputs, with batch sizes 64 and 1 for throughput and latency, respectively. RPAttention has lower throughput at window size 7 and comparable throughput at 14. At sizes 28 and 56, it uses 17.2% and 43.9% fewer FLOPs and achieves $1.89\times$ and $3.89\times$ the throughput of Swin-T, respectively. This comparison measures execution cost, not accuracy equivalence.

D COMPARISONS WITH OTHER METHODS

We provide additional comparisons with FLatten (Han et al., 2023), Agent Attention (Han et al., 2024b), PolaFormer (Meng et al., 2025), InLine (Han et al., 2024a), MobiAtt (Yao et al., 2024), and CA (Han et al., 2026). Gray shading identifies RPAttention, and boldface indicates the best task performance within each backbone group.

D.1 IMAGENET-1K CLASSIFICATION

Table D.1 compares classification accuracy, parameters, and FLOPs across four backbone families. RPAttention achieves the highest reported Top-1 accuracy among the listed methods in each backbone group, tying with MobiAtt on PVTv2-B2.

Table D.1: ImageNet-1K classification comparison with other efficient-attention methods. Parentheses indicate Top-1 gains in percentage points over the corresponding backbone. Best Top-1 accuracy within each group is highlighted in bold.

Method	Params	FLOPs	Top-1
DeiT-T	5.7M	1.2G	72.2
+ FLatten (2023)	6.1M	1.1G	74.1 (+1.9)
+ Agent (2024b)	6.0M	1.2G	74.9 (+2.7)
+ PolaFormer (2025)	6.1M	1.2G	74.6 (+2.4)
+ InLine (2024a)	6.5M	1.1G	74.5 (+2.3)
+ RPAttention	5.9M	1.2G	**75.4 (+3.2)**
DeiT-S	22.1M	4.6G	79.8
+ Agent (2024b)	22.7M	4.4G	80.5 (+0.7)
+ InLine (2024a)	16.7M	5.0G	80.2 (+0.4)
+ RPAttention	22.3M	4.5G	**80.8 (+1.0)**
PVT-T	13.2M	1.9G	75.1
+ FLatten (2023)	12.2M	2.0G	77.8 (+2.7)
+ Agent 2024b	11.6M	2.0G	78.4 (+3.3)
+ PolaFormer (2025)	12.0M	2.0G	78.8 (+3.7)
+ InLine 2024a	12.0M	2.0G	78.2 (+3.1)
+ RPAttention	11.5M	2.2G	**79.2 (+4.1)**
PVT-S	24.5M	3.8G	79.8
+ FLatten (2023)	21.7M	4.0G	81.7 (+1.9)
+ Agent 2024b	20.6M	4.0G	82.2 (+2.4)
+ PolaFormer (2025)	21.0M	4.1G	81.9 (+2.1)
+ InLine 2024a	21.6M	3.9G	82.0 (+2.2)
+ RPAttention	20.5M	4.4G	**82.5 (+2.7)**
PVT-M	44.2M	6.7G	81.2
+ FLatten (2023)	37.2M	7.0G	83.0 (+1.8)
+ Agent (2024b)	35.9M	7.0G	83.4 (+2.2)
+ InLine (2024a)	37.6M	6.9G	83.2 (+2.0)
+ RPAttention	35.5M	7.5G	**83.5 (+2.3)**

Method	Params	FLOPs	Top-1
PVTv2-B0	3.7M	0.5G	70.5
+ FLatten (2023)	3.6M	0.6G	71.1 (+0.6)
+ MobiAtt (2024)	3.5M	0.6G	71.5 (+1.0)
+ PolaFormer (2025)	3.4M	0.6G	72.3 (+1.8)
+ RPAttention	3.2M	0.7G	**73.1 (+2.6)**
PVTv2-B1	13.1M	2.1G	78.7
+ FLatten (2023)	12.9M	2.2G	79.5 (+0.8)
+ PolaFormer (2025)	13.0M	2.2G	80.2 (+1.5)
+ RPAttention	12.3M	2.4G	**80.5 (+1.8)**
PVTv2-B2	25.4M	4.0G	82.0
+ FLatten (2023)	22.6M	4.3G	82.5 (+0.5)
+ MobiAtt (2024)	21.1M	3.8G	**82.6 (+0.6)**
+ RPAttention	21.3M	4.6G	**82.6 (+0.6)**
Swin-T	29.0M	4.5G	81.3
+ FLatten (2023)	29.0M	4.5G	82.1 (+0.8)
+ Agent (2024b)	29.0M	4.5G	82.6 (+1.3)
+ PolaFormer (2025)	29.0M	4.5G	82.6 (+1.3)
+ InLine (2024a)	30.0M	4.5G	82.4 (+1.1)
+ CA (2026)	28.0M	4.6G	82.2 (+0.9)
+ RPAttention	28.4M	4.9G	**82.7 (+1.4)**
Swin-S	50M	8.7G	83.0
+ FLatten (2023)	51M	8.7G	83.5 (+0.5)
+ Agent (2024b)	50M	8.7G	83.7 (+0.7)
+ PolaFormer (2025)	50M	8.7G	83.6 (+0.6)
+ InLine (2024a)	50M	8.7G	83.6 (+0.6)
+ CA (2026)	50M	8.8G	83.6 (+0.6)
+ RPAttention	50M	9.3G	**83.8 (+0.8)**

D.2 DENSE PREDICTION

Table D.2 compares RPAttention with other efficient attention methods on COCO and ADE20K. RPAttention consistently ranks among the top two on the reported AP, mIoU, and mAcc metrics within each backbone group. On COCO, it achieves the best or tied-best box AP and the best mask AP across all three PVT variants. On ADE20K, it achieves the highest mIoU in most backbone groups. These results highlight its strong performance across dense prediction tasks.

Table D.2: Comparison with other efficient-attention methods on COCO and ADE20K. COCO results use Mask R-CNN with the $1\times$ schedule. Unavailable entries are denoted by "–". Bold and underlined values indicate the best and second-best distinct scores within each backbone group, respectively.

Method	**Detection & Instance Seg.**							**Semantic Seg.**		
	FLOPs	$\mathbf{AP}^b$	$\mathbf{AP}^b_{50}$	$\mathbf{AP}^b_{75}$	$\mathbf{AP}^m$	$\mathbf{AP}^m_{50}$	$\mathbf{AP}^m_{75}$	**FLOPs**	**mIoU**	**mAcc**
PVT-T	240G	36.7	59.2	39.3	35.1	56.7	37.3	158G	36.57	46.72
+ FLatten (2023)	244G	<u>38.2</u>	61.6	41.9	37.0	57.6	39.0	169G	37.21	48.95
+ PolaFormer (2025)	216G	**40.4**	<u>62.4</u>	**43.9**	<u>37.4</u>	<u>59.4</u>	<u>40.3</u>	133G	38.3	–
+ InLine (2024a)	–	–	–	–	–	–	–	127G	<u>39.16</u>	<u>50.63</u>
+ RPAttention	215G	**40.4**	**62.8**	<u>43.8</u>	**37.7**	**59.9**	**40.4**	128G	**40.00**	**53.54**
PVT-S	305G	40.4	62.9	43.8	37.8	60.1	40.3	225G	41.95	53.02
+ Agent (2024b)	–	–	–	–	–	–	–	211G	**44.18**	**56.17**
+ PolaFormer (2025)	262G	<u>43.9</u>	66.1	<u>47.9</u>	<u>40.2</u>	<u>63.1</u>	43.0	179G	41.0	–
+ InLine (2024a)	250G	43.4	<u>66.4</u>	47.1	40.1	<u>63.1</u>	<u>43.3</u>	168G	42.93	54.58
+ RPAttention	259G	**44.1**	**66.6**	**48.5**	**40.8**	**63.7**	**44.3**	177G	<u>43.25</u>	<u>56.04</u>
PVT-M	392G	42.0	64.4	45.6	39.0	61.6	42.1	315G	42.91	53.80
+ Agent (2024b)	–	–	–	–	–	–	–	321G	44.30	56.42
+ InLine (2024a)	310G	<u>44.0</u>	<u>66.4</u>	<u>48.0</u>	<u>40.3</u>	<u>63.4</u>	<u>43.5</u>	229G	<u>44.59</u>	<u>57.20</u>
+ RPAttention	322G	**45.9**	**68.2**	**50.4**	**41.9**	**65.4**	**45.3**	241G	**45.16**	**58.13**
Swin-T	267G	43.7	66.6	47.7	<u>39.8</u>	63.3	42.7	945G	44.51	55.61
+ FLatten (2023)	–	–	–	–	–	–	–	946G	44.82	57.01
+ Agent (2024b)	261G	41.4	63.5	45.2	38.9	60.2	40.2	954G	<u>46.68</u>	<u>58.53</u>
+ PolaFormer (2025)	272G	**44.8**	**67.6**	**49.1**	**40.5**	<u>64.1</u>	**43.5**	946G	45.8	–
+ InLine (2024a)	–	–	–	–	–	–	–	941G	45.57	57.60
+ RPAttention	271G	<u>44.5</u>	<u>67.3</u>	<u>48.4</u>	**40.5**	**64.2**	<u>43.3</u>	949G	**46.97**	**59.17**
Swin-S	358G	45.7	67.9	<u>50.4</u>	41.1	64.9	44.2	1038G	47.64	58.78
+ FLatten (2023)	–	–	–	–	–	–	–	1038G	<u>48.14</u>	59.31
+ Agent (2024b)	364G	**47.2**	**69.6**	**52.3**	**42.7**	**66.6**	**45.8**	1043G	48.08	<u>59.78</u>
+ RPAttention	360G	<u>46.3</u>	<u>69.0</u>	<u>50.4</u>	<u>41.8</u>	<u>65.7</u>	<u>45.1</u>	1042G	**48.48**	**60.12**

E Additional Analyses and Visualizations

E.1 Cross-Seed Routing Stability

We examine whether RPAttention learns consistent spatial routing across independent training runs. Two RPAttention-PVTv2-B1 models are trained with identical settings using random seeds 123 and 456. To compare their routing independently of representative ordering, we use the composed query-to-space response in Eq. 16.

$$R_{q,\text{full}} = \text{Normalize}_{\text{space}} \left[\frac{1}{H} \sum_{h=1}^{H} D_{q,:}^{(h)} \left(I + S^{(h)} \right) \hat{A}^{(h)\top} \right]. \tag{16}$$

where H is the number of attention heads, I is the identity matrix, and $\hat{A}$, S, and D denote the normalized Gather assignment, Interact attention, and Distribute attention, respectively. The operator $\text{Normalize}_{\text{space}}$ normalizes each response over spatial locations.

The composed response in Eq. 16 is invariant to the ordering of representatives. Under an arbitrary representative permutation P, the corresponding quantities transform as $\hat{A}' = \hat{A}P^\top$, $D' = DP^\top$, and $S' = PSP^\top$. Consequently,

$$\begin{aligned} D' \left(I + S' \right) \hat{A}'^\top &= DP^\top \left(I + PSP^\top \right) P\hat{A}^\top \\ &= D \left(I + S \right) \hat{A}^\top. \end{aligned} \tag{17}$$

Equation 17 shows that the representative permutations cancel in the composed spatial response.

Table E.1: Cross-seed stability of composed query-to-space responses for matched image–query pairs on 1,000 COCO-val images. Metrics are averaged over queries, and brackets report 95% confidence intervals obtained by bootstrap resampling images.

Stage	**Cosine ↑**	**JS ↓**	**Top-20% IoU ↑**
Stage 3	0.9491 [0.9484, 0.9497]	0.0125 [0.0123, 0.0126]	0.5421 [0.5382, 0.5460]
Stage 4	0.9610 [0.9601, 0.9619]	0.0098 [0.0096, 0.0100]	0.4860 [0.4795, 0.4926]

On 1,000 COCO-val images evaluated at the same nine query locations in the last blocks of Stages 3 and 4, the independently trained models exhibit high agreement in their composed response distributions, with cosine similarities above 0.94 and low JS divergences (Table E.1). The Top-20% IoU values are more moderate, indicating that the strongest-response regions are similar but not identical across training runs.

As an additional control, we evaluate the direct Gather–Distribute response at Stage 3 by replacing $(I + S^{(h)})$ with I in Eq. 16. This excludes representative interaction from the response composition while preserving invariance to representative ordering. The cross-seed cosine similarity is 0.9107 for the same image and query, compared with 0.6470 for the same image with different queries and 0.6154 for different images at the same query coordinates. These results support input- and query-dependent routing consistency that cannot be attributed solely to mixing through Interact.

E.2 Foreground Alignment Analysis

On 1,000 class-stratified ImageNet validation images with localization annotations, we analyze Agent Attention–PVT-S and RPAttention–PVT-S using identical preprocessing. We measure: (1) *foreground concentration* (attention mass in the ground-truth region divided by its area fraction; uniform attention scores 1); (2) *pointing accuracy* (whether the response maximum lies in a ground-truth box); and (3) *map redundancy* (mean pairwise cosine similarity among representative maps). Each entry in Table E.2 reports Agent Attention / RPAttention. RPAttention achieves higher foreground concentration and pointing accuracy, together with lower map redundancy, at every stage (Table E.2).

All paired differences have image-level bootstrap 95% confidence intervals excluding zero. At Stage 4, RPAttention also exhibits a higher normalized effective rank (0.247 vs. 0.193) and greater adjacent-layer consistency (0.768 vs. 0.720), while maintaining comparable flip consistency (0.897 vs. 0.901). Together, these observations indicate that RPAttention forms more distinct representative maps with stronger foreground alignment and consistent responses across layers and horizontal flips.

Table E.2: Foreground alignment of intermediate tokens (Agent Attention / RPAttention on PVT-S, ImageNet-1K validation).

Stage	Foreground concentration ↑	Pointing accuracy ↑	Map redundancy ↓
1	0.961 / **1.115**	0.568 / **0.649**	0.378 / **0.089**
2	0.999 / **1.146**	0.571 / **0.660**	0.395 / **0.242**
3	1.226 / **1.325**	0.643 / **0.662**	0.260 / **0.230**
4	0.947 / **1.039**	0.491 / **0.529**	0.807 / **0.532**

E.3 Additional Query-Response Visualizations

Figure E.1 provides additional examples of how RPAttention connects spatial queries with image content, complementing the comparisons in Figure 3.

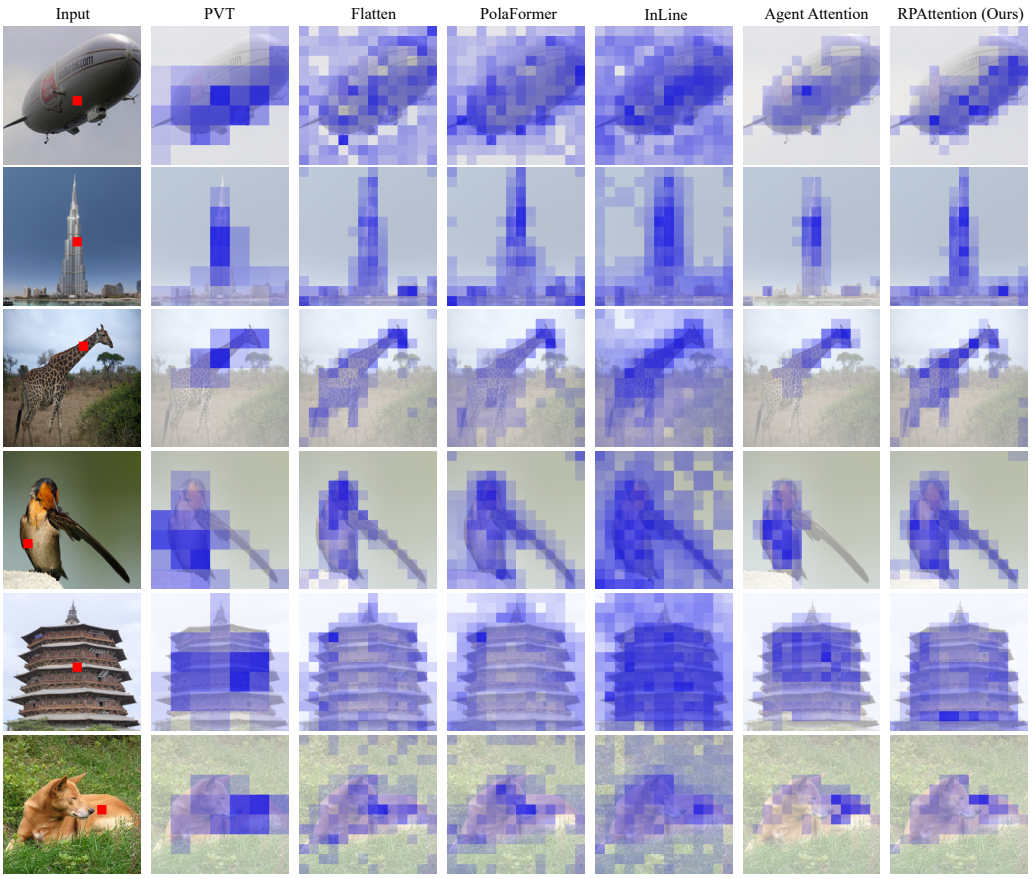

Figure E.1: Additional query-response visualizations. For selected query locations, RPAttention produces response maps that align with semantically related image regions, illustrating how representatives mediate global information exchange beyond fixed spatial layouts.