DO VISION TRANSFORMERS NEED ALL-TO-ALL ATTENTION? GLOBAL COMMUNICATION THROUGH ELASTIC LEARNED CORES

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision Transformers (ViTs) learn rich visual-semantic representations through all-to-all self-attention among patch tokens. However, this design implicitly assumes that direct pairwise patch interactions are necessary for effective representation learning. In this work, we challenge this assumption and show that representations supporting both global recognition and dense prediction can be learned without direct patch-to-patch interaction. We propose **VECA** (Visual Elastic-Core Attention), a vision transformer with core-periphery structured attention mediated by a small set of learned cores. Patch tokens exchange global information exclusively through these cores, while the full set of dense patches are preserved and iteratively updated across layers. This reduces attention complexity from $O(N^2)$ to $O(N)$, linear in the number of patches N for a fixed core budget C. Unlike prior latent-token cross-attention architectures, VECA facilitates sparse global communication without compressing the spatial representation itself. Nested training along the core axis further enables a single model to elastically trade off computation and accuracy at inference time without retraining. Across image classification and dense prediction tasks, VECA remains competitive with full-attention backbones and outperforms the evaluated linear-complexity alternatives on most benchmarks. Moreover, without explicit supervision, these cores develop semantically organized structures that support object-label transfer across views. These results show that effective visual representations can be learned without direct all-to-all patch interaction.

1 INTRODUCTION

Recent advances in computer vision have been driven by increases in computational resources and dataset scale. These developments have enabled a shift away from strong inductive biases and towards more flexible, data-driven models (Tolstikhin et al., 2021; Touvron et al., 2022; Liu et al., 2021; Lee-Thorp et al., 2022; Rao et al., 2023). Whereas convolutional neural networks directly encode locality and translation equivariance into their structure (Luo et al., 2016; Islam et al., 2020), these design choices constrain the representations that can be learned at scale. As datasets have grown, the value of these inductive biases has diminished. Vision Transformers (ViTs, Dosovitskiy et al. (2020)) rely on self-attention with only weak spatial constraints from patchification, enabling global receptive fields at every layer (Raghu et al., 2021; Park & Kim, 2022). In standard ViTs, this global communication is realized through dense token-to-token interaction, where every patch attends to every other patch. While effective, these all-to-all interactions incur *quadratic complexity*, becoming increasingly costly at high resolutions (Li et al., 2023; Nauen et al., 2025). Latent-mediated cross-attention offers an efficient alternative, but compressing inputs into a small set of latents can discard the spatially aligned representations maintained by self-attention. This motivates a fundamental question: *Do Vision Transformers require all-to-all patch interactions to learn effective visual representations?*

In this work, we introduce Visual Elastic-Core Attention (**VECA**), which replaces standard self-attention with a block-sparse attention pattern mediated by a small, resolution-invariant set of C learned core tokens. Viewed as an attention connectivity graph, VECA forms a *core-periphery network* (Rombach et al., 2014; Zhang et al., 2015): the cores form a fully connected clique (Luce & Perry, 1949) that interacts with all image patches, while patches never directly attend to one another. Information is therefore exchanged through the cores while all N dense patch representations are

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.

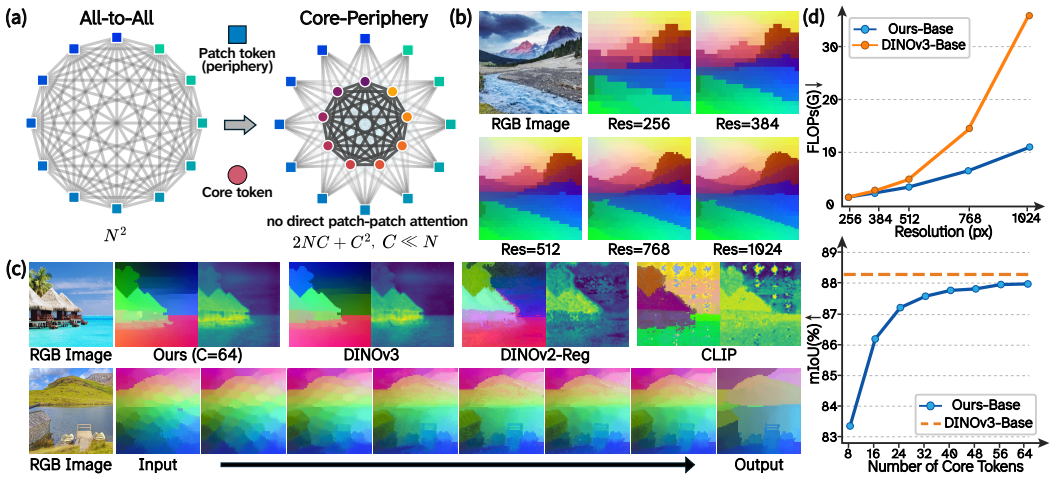

Figure 1: **VECA overview and learned representations.** **(a)** Standard attention utilizes a fully connected attention matrix with N^2 comparisons given N input patches. **VECA** constructs a core-periphery matrix with C core tokens that form a clique, eliminating dense patch-to-patch interaction and requiring only $2NC + C^2$ comparisons. **(b)** Dense feature visualization with UMAP. Our method produces stable embeddings across resolutions. **(c) Top**: We visualize our features and `[CLS]`-patch cosine similarity of our model, comparing the visualizations to its teacher (DINOv3) as well as other baseline embeddings. **Bottom**: We visualize the cores' attention maps to patches (Appendix A.9) with UMAP. The maps start visually isotropic (spherical) and become increasingly semantic with depth. **(d) Top**: FLOPs of our attention block versus DINOv3 across resolutions. **Bottom**: Elastically varying the number of active core tokens, trading off accuracy and speed in PASCAL VOC.

preserved and iteratively updated across layers. The resulting attention cost is $(2NC + C^2)$, which is linear in N for a fixed core budget C. VECA's core-based design also naturally supports nested dropout (Rippel et al., 2014) along the core axis. This allows the number of active cores to vary at inference time and enables a single model to elastically trade computation for accuracy.

Following recent work, we explore our proposed architecture by *distilling* representations from strong vision foundation models (Wei et al., 2022; Ranzinger et al., 2024; Shang et al., 2024; Sarıyıldız et al., 2024). Specifically, we train VECA using a DINOv3 (Siméoni et al., 2025) teacher. Across image classification and dense prediction tasks, VECA remains competitive with full-attention backbones and outperforms the evaluated alternatives on most benchmarks. Notably, at 512×512 resolution, VECA-B/16 with only $C = 8$ cores retains 95.7% of the PASCAL VOC (Everingham et al., 2015) performance of the full-attention ViT-B/16 while using only 1.6% as many attention interactions. We illustrate the architecture in Figure 1. We also observe *emergent* structure in the cores, which develop semantically aligned organization without explicit supervision and support label transfer from a reference frame to later video frames. Our contributions are therefore threefold: **(1)** a core-periphery attention architecture that removes direct patch-to-patch interaction while preserving and continually updating dense patch representations; **(2)** nested core training that enables elastic computation at inference time; and **(3)** an analysis showing that repeated interaction with the updated dense patch tokens gives rise to structured core representations with semantic organization that supports object-label transfer across frames of the same video. Together, our results show that effective visual representations can be learned without all-to-all patch interaction.

2 Related Work

Efficient and Latent-Mediated Attention. To mitigate the quadratic cost of self-attention, prior work approximates softmax (Shen et al., 2021; Katharopoulos et al., 2020; Choromanski et al., 2020; Zhang et al., 2024a; Joshi et al., 2025; Lu et al., 2026), factorizes attention at low rank (Wang et al., 2020; Xiong et al., 2021), restricts it to hashed, sparse, or sliding-window patterns (Kitaev et al., 2020; Zaheer et al., 2020; Beltagy et al., 2020; Roy et al., 2021; Acharya et al., 2024), or replaces it with unparameterized transforms (Lee-Thorp et al., 2022; Tay et al., 2021). Linear RNNs offer an alternative for sequences (Peng et al., 2021; Irie et al., 2021; Gu et al., 2022; Peng et al., 2022; Mao,

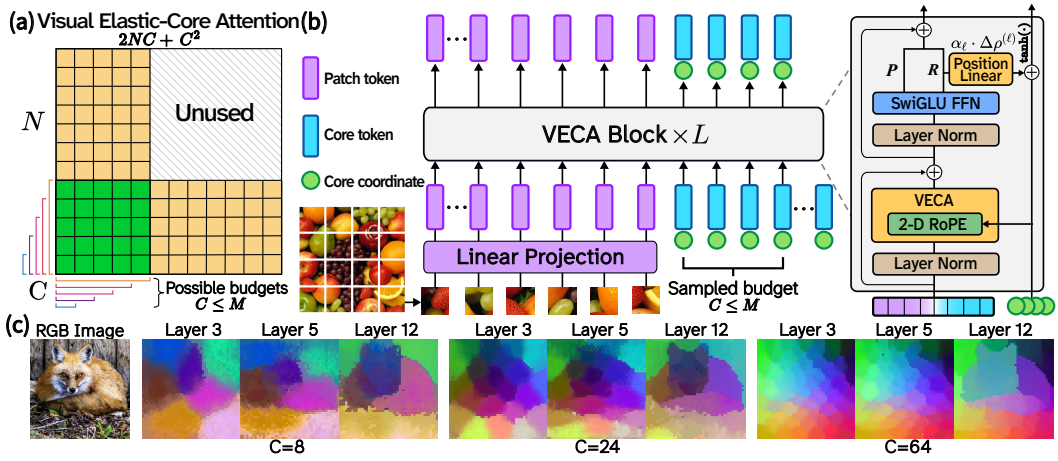

Figure 2: **Architecture of VECA. (a)** Core-periphery attention: cores attend to all tokens, patches attend only to the C active cores, and patch-to-patch attention is unused, yielding $2NC + C^2$ interactions and graph diameter 2. Budgets $C \leq M$ are nested prefixes of a single core bank. **(b)** VECA preserves N dense patch tokens alongside a sampled budget of C core tokens; both are updated at every layer, together with learned core coordinates used in 2-D RoPE. **(c)** Joint UMAP across layers of the cores' attention to patches. Fewer cores summarize broader regions, larger budgets yield finer partitions, and semantic organization emerges with depth.

2022; Yang et al., 2023; Sun et al., 2023; Gu & Dao, 2023; Qin et al., 2024; Zhang et al., 2024c; Beck et al., 2024; Peng et al., 2025; Hu et al., 2025), and linear attention has been applied to vision with some performance tradeoffs (Bolya et al., 2022b; Cai et al., 2024a; You et al., 2023; Han et al., 2023; Meng et al., 2025). Latent-mediated designs instead route communication through a small token set: Set Transformer and Agent Attention use layer-local latent tokens (Lee et al., 2019; Han et al., 2024), Luna packs inputs into a fixed-length sequence via nested attention (Ma et al., 2021), Perceiver compresses inputs into a latent array (Jaegle et al., 2021; 2022; Dolga et al., 2025; Jajal et al., 2026), and bottleneck tokens mediate multimodal fusion (Nagrani et al., 2021). VECA retains exact softmax attention and obtains linear cost from a sparse core-periphery graph. Unlike Perceiver, it updates the full patch stream at every layer; unlike Set Transformer and Agent Attention, its cores persist across depth and share a single attention operator with the patches. VECA is also natively elastic, enabling test-time tradeoffs between compute and accuracy.

Elastic and Adaptive Computation. Elastic inference trades computation for accuracy at test time (Zilberstein, 1996). Matryoshka representations learn truncatable embeddings (Kusupati et al., 2022), and nested dropout learns ordered parameter hierarchies (Rippel et al., 2014; Staley & Markowitz, 2022; Shen & Meinshausen, 2024; Ho et al., 2025; Ladjal et al., 2019), later extended to width-adaptive networks (Tann et al., 2016; Kim et al., 2018; Yu et al., 2018) and sub-network extraction (Cai et al., 2019; Valipour et al., 2023; Li & Luo, 2024; Zhang et al., 2024b). Transformer-specific designs vary model width (Devvrit et al., 2024; Cai et al., 2024b), whereas VECA uses core-based elasticity. Input-dependent computation uses early exiting (Teerapittayanon et al., 2016; Wu et al., 2018; Liu et al., 2020) or iterative processing (Graves, 2016; Dehghani et al., 2018; Xie et al., 2025; Jeddi et al., 2026), while data-level elasticity uses token merging (Bolya et al., 2022a), pruning of spatially redundant tokens (Yin et al., 2022; Rao et al., 2021), scratchpads (Xue et al., 2023), or iterative expansion (Hojjat et al., 2025). Sparse experts offer another axis (Gu et al., 2025; Wang et al., 2025). Closest to our work, MQT learns elastic queries (Hu et al., 2024), but as a single layer on a frozen, quadratic-time CLIP model. In contrast, VECA is an elastic *visual backbone* competitive with state-of-the-art vision foundation models on both *classification* and *dense tasks*.

Differentiable Clustering and Structured Representations. Learnable clustering jointly optimizes representations and their groupings, using discrete assignment (Xie et al., 2016; Yang et al., 2017; Duan et al., 2019) or continuous relaxations (Shah & Koltun, 2018; Bianchi et al., 2020; Saha et al., 2023; Stewart et al., 2023), with unknown cluster counts inferred via Bayesian methods, density estimation, or pruning (Ronen et al., 2022; Ren et al., 2020; Comaniciu & Meer, 2002; Kong & Fowlkes, 2018; Leiber et al., 2021; 2024). For images, superpixels (Achanta et al., 2012;

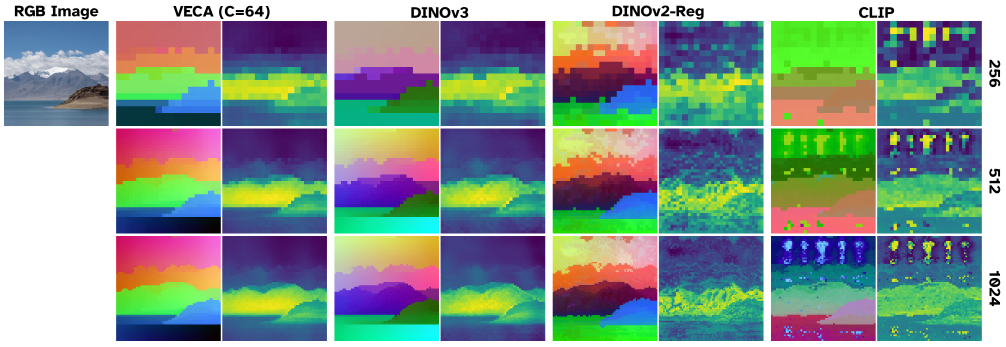

Figure 3: **Learned dense representations across resolutions.** UMAP visualizations (**left**) and `[CLS]`-dense cosine-similarity maps (**right**) show that VECA preserves coherent and semantically organized structure as resolution increases, despite removing direct patch-to-patch attention.

Jampani et al., 2018) and region-aware tokenization (Aasan et al., 2024; Mei et al., 2024; Aasan et al., 2025) partition images into content-adaptive regions, token aggregation methods (Bolya et al., 2022a; Aasan et al., 2024; Mei et al., 2024; Aasan et al., 2025; Khosla et al., 2025; Pu et al., 2025) compress dense tokens for efficiency, and slot attention (Locatello et al., 2020) and clustering-based transformers (Xu et al., 2022; Ke et al., 2022; Liang et al., 2023; Zeng et al., 2024; Brasó et al., 2025) organize features into coherent regions. VECA's cores can likewise be interpreted as cluster seeds that become increasingly semantic with depth (Section 4.4). However, rather than merging tokens or iteratively refining slots, VECA preserves every patch token and refines cores and patches jointly in a single feed-forward pass, allowing it to serve as a vision backbone with elastic, linear-cost inference.

3 Methods

In this section, we introduce our **VECA** (**V**isual **E**lastic-**C**ore **A**ttention) architecture, illustrated in Figure 2. Unlike standard Vision Transformers, which compute full self-attention, VECA organizes attention around a block-sparse core-periphery structure (Rombach et al., 2014; Zhang et al., 2015). Image patches form the "periphery," and a small set of learned "cores" forms a fully connected communication interface. Patch tokens are updated throughout the network but exchange information only through the cores, preserving spatially aligned dense representations that remain coherent as input resolution increases (Figure 3). Section 3.1 reviews the relevant background on Vision Transformers. We then introduce the VECA module and its key properties in Section 3.2, followed by the budget-adaptive training mechanism in Section 3.3.

3.1 Background on Vision Transformers

Here, we go over the fundamental components of Vision Transformers and outline recent advancements. Given an RGB image $\mathcal{I} \in \mathbb{R}^{H\times W\times 3}$, the image is divided into a grid of non-overlapping $N = (H/P) \times (W/P)$ spatial patches, each with width P. Each patch is flattened and linearly projected to a latent dimension D. Positional embeddings can be added to these projections to retain spatial coordinates, resulting in a 1D sequence $\mathbf{X} \in \mathbb{R}^{N\times D}$.

The sequence $\mathbf{X}$ is then passed through Transformer blocks driven by self-attention. The input is linearly projected into Query ($\mathbf{Q}$), Key ($\mathbf{K}$), and Value ($\mathbf{V}$) matrices. The self-attention mechanism computes contextualized representations by taking a similarity-weighted sum of the values:

$$\text{Self-Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}, \tag{1}$$

where d_k is the dimension of the keys. Typically, an additional `[CLS]` token and N_R register tokens (Burtsev et al., 2020; Darcet et al., 2023; Chen et al., 2025b) are concatenated to the image tokens, creating a total attention matrix of size $(N + N_R + 1)^2$, quadratic in the number of image patches. In practice, models employ Multi-Head Self-Attention, executing this operation in parallel and concatenating the outputs. While the original ViT used learnable positional embeddings that were adapted to higher resolutions via interpolation, recent work has adopted 2-D axial RoPE (Fang

Table 1: **Frozen-backbone evaluation across global and dense visual tasks.** All models use approximately 86M parameters and 16×16 patches unless noted; ImageNet-1K top-1 is evaluated at 256 (252 for patch-14 models) and dense tasks at 512 (518). Bold marks the best linear-complexity result. Controlled models are trained with the same protocol; among them, VECA leads the linear-complexity architectures on six of seven benchmarks and remains close to the full-attention ViT-B/16.

Method	Params (M)	IN1K ↑	Segmentation (mIoU ↑)				Depth (RMSE ↓)	
			VOC	Context	ADE	Stuff	NYUv2	KITTI
External pretrained ViT references								
CLIP-B/16 (Radford et al., 2021)	86.2	79.33	72.87	45.99	34.55	36.18	0.6176	4.3499
DINOv2-reg-B/14 (Darcet et al., 2023)	86.6	83.44	84.23	55.02	48.92	45.42	0.3953	2.8307
DINOv3-B/16 (Teacher for all models below)	85.6	83.56	87.50	57.74	51.35	48.41	0.3684	2.6915
Controlled quadratic softmax baseline								
ViT-B/16 (Dosovitskiy et al., 2020)	85.7	82.30	87.60	58.01	50.94	48.49	0.3683	2.6121
Latent/inducing-token linear architectures								
Set Transformer (Lee et al., 2019)	85.7	80.15	85.87	56.07	48.34	46.28	0.3958	2.8072
Perceiver IO (Jaegle et al., 2022)	86.1	79.49	79.05	51.18	42.50	41.23	0.4418	3.5240
Linear-attention architectures								
Efficient Attention (Shen et al., 2021)	85.7	81.24	86.40	56.85	49.47	47.13	0.3836	2.7087
Flowformer (Wu et al., 2022)	85.7	81.11	86.69	56.60	50.01	46.94	0.3857	2.7355
Hydra Attention (Bolya et al., 2022b)	85.7	79.12	83.59	53.46	43.60	43.20	0.4362	2.9828
PolaFormer (Meng et al., 2025)	85.7	81.85	86.74	57.27	50.07	47.54	0.3792	**2.6718**
Landmark-based linear approximation								
Nyströmformer (Xiong et al., 2021)	85.7	81.34	86.48	56.86	50.08	47.26	0.3843	2.7949
Ours								
VECA-B/16 ($C = 64$)	85.7	**82.18**	**87.07**	**57.46**	**50.69**	**47.92**	**0.3705**	2.7252

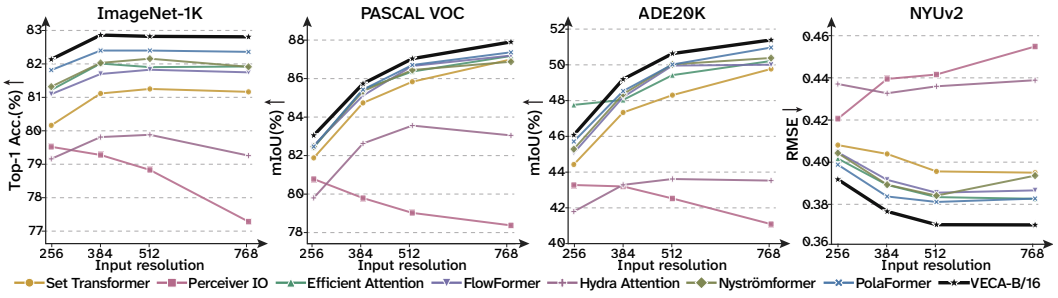

Figure 4: **Representation quality across input resolutions.** We compare VECA-B/16 with linear-complexity baselines from 256 to 768 on classification, segmentation, and depth. VECA improves with resolution and leads all baselines at 768, whereas Perceiver IO degrades as resolution increases.

et al., 2024; Lu et al., 2024a;b; Siméoni et al., 2025). Following DINOv3, we adopt 2-D axial RoPE in our model, where core tokens have content- and layer-dependent learnable positions.

3.2 Core-Mediated Visual Attention

The basic building block of VECA is a block-sparse attention operation between active core tokens and patch tokens. Given patch tokens $Z = \{z_1, \ldots, z_N\}$, we additionally maintain an ordered bank of learnable core tokens $R_M = \{r_1, \ldots, r_M\}$, where M denotes the maximum core capacity. For each image, we select an active budget $C \leq M$. The input sequence is formed by concatenating the active core prefix $R_C = R_M[:C]$ with the patch sequence, $X = [R_C; Z]$. Unlike standard self-attention, patch tokens do not attend directly to all other patch tokens. Instead, core tokens attend to the full sequence, allowing them to aggregate global information, while patch tokens attend only to the active cores, allowing them to receive global context through the core interface. In standard query-key-value notation, the block computes

$$R' = \mathrm{Attn}(R_C, X, X), \qquad Z' = \mathrm{Attn}(Z, R_C, R_C), \tag{2}$$

which are concatenated to form $X' = [R'; Z']$, where we use 2-D axial RoPE.

This gives the attention matrix a core-periphery structure: fully connected core tokens serve as a compact global communication interface, while patch tokens retain spatially aligned dense representations.

Table 2: **Ablation of VECA's core-periphery design.** We compare VECA with latent-token architectures that lack dense patch updates and core-patch interaction (Perceiver IO, VECA-CoreOnly) and with a variant using fixed core positions. FLOPs are for a 256×256 forward pass; colored values are differences from VECA-B at $C = 64$. Removing both causes the largest drop on every task.

Variant	IN1K Acc. ↑	ADE mIoU ↑	NYUv2 RMSE ↓	FLOPs (G)	Params (M)
No patch updates or core-patch interaction					
Perceiver IO (256 latents) (Jaegle et al., 2022)	$79.49_{-2.69}$	$42.50_{-8.19}$	$0.4418_{+0.0713}$	23.36	86.06
VECA-CoreOnly ($C = 64$)	$71.08_{-11.10}$	$40.49_{-10.20}$	$0.4584_{+0.0879}$	9.39	85.73
VECA-CoreOnly ($C = 256$)	$80.30_{-1.88}$	$49.10_{-1.59}$	$0.3857_{+0.0152}$	27.13	85.87
No core coordinate updates					
Fixed core coordinates ($C = 64$)	$81.71_{-0.47}$	$49.80_{-0.89}$	$0.3771_{+0.0066}$	28.01	85.71
Full VECA architecture					
VECA-B ($C = 8$)	$80.45_{-1.73}$	$44.51_{-6.18}$	$0.4330_{+0.0625}$	22.65	85.73
VECA-B ($C = 64$)	**82.18**	**50.69**	**0.3705**	28.01	85.73

Table 3: **Ablation of nested elastic training.** At $C = 64$, nested elastic training matches or slightly exceeds fixed-budget training on classification, segmentation, and depth, so elasticity costs no full-budget performance.

Training scheme	**IN1K Acc. ↑**	**ADE mIoU ↑**	**NYUv2 RMSE ↓**
Fixed-budget training	$82.14_{-0.04}$	$50.66_{-0.03}$	$0.3713_{+0.0008}$
VECA Nested elastic training	**82.18**	**50.69**	**0.3705**

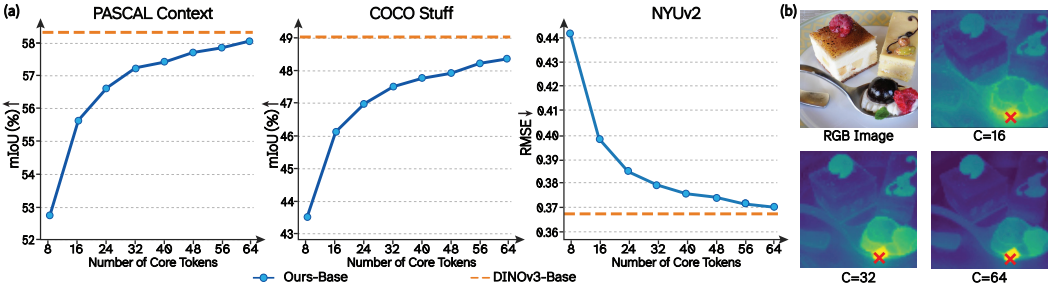

Figure 5: **Elastic computation through the core-token budget.** **(a)** Frozen-backbone dense prediction at 768 px as the active core budget varies from $C = 8$ to $C = 64$. Performance improves smoothly with additional cores across multiple datasets. **(b)** Patch-feature similarity across core budgets. Larger budgets produce finer and more spatially localized semantic organization.

The resulting attention cost is $(2NC + C^2)$, compared with N^2 (ignoring `[CLS]` and registers) for dense patch self-attention. Therefore, for fixed C, VECA attention cost scales linearly in the number of patches N while still allowing information exchange across the full image. This core-periphery structure has graph diameter 2, compared with diameter 1 for full self-attention, and therefore requires two blocks for information from one patch to influence another patch through the core interface. As we demonstrate in Section 4, VECA nevertheless learns competitive representations under this restricted communication structure.

Learned Core Coordinates. Each core token is associated with a continuous image- and layer-dependent planar coordinate in $[-1, 1] \times [-1, 1]$, while patch tokens use fixed image-grid coordinates. Core and patch tokens are modulated with two-dimensional rotary positional encoding. Core coordinates are initialized to cover the image plane and are updated across layers. For layer ℓ and core i, where r_i^ℓ is the feature of core token i, $\rho_i^\ell \in \mathbb{R}^2$ is the unconstrained coordinate, $u_i^\ell \in [-1, 1]^2$ is the coordinate used for RoPE, f_ℓ is a lightweight coordinate head, and α_ℓ is a learned layer-wise scalar:

$$\rho_i^{\ell+1} = \rho_i^\ell + \alpha_\ell f_\ell(r_i^\ell), \qquad u_i^{\ell+1} = \tanh(\rho_i^{\ell+1}). \tag{3}$$

This allows each core to maintain both a semantic representation and an evolving spatial position.

3.3 Budget-Adaptive Representation Learning

VECA is trained to support multiple active-core budgets within a single shared model. The ordered core bank allows us to select prefixes of variable size (Rippel et al., 2014; Hu et al., 2024). We define a distribution $p_C(\cdot)$ over active core budgets. During training, we sample $C \sim p_C(\cdot)$ and retrieve the active prefix $R_C = R_M[:C] = (r_1, \ldots, r_C)$. Since the sampled cores form a prefix, early cores are

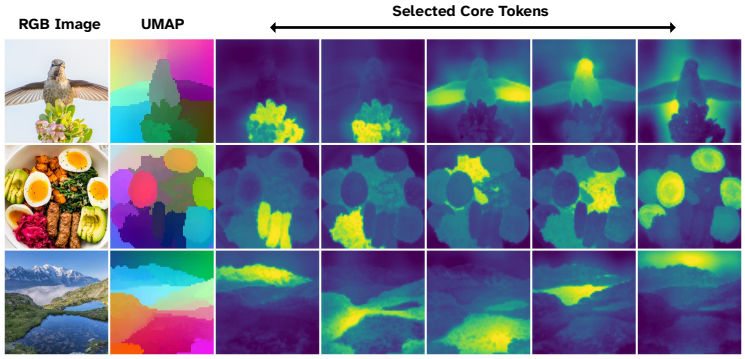

Figure 6: **Semantic roles of core tokens.** We visualize cosine similarity between selected core tokens and spatial patch features. Brighter colors indicate higher similarity. Cores localize to different objects and parts, suggesting that the cores develop distinct yet complementary semantic roles.

active under more budgets and are encouraged to encode the most broadly useful information, while later cores provide additional capacity for higher-fidelity representations.

This nested design encourages a spatially and semantically coarse-to-fine organization of the core representation. With a small core budget, global information must be communicated through a compact set of cores, while the full dense patch representation is still preserved. As the budget increases, additional cores can specialize to more spatially local or semantically specific information. At inference time, a single VECA model can trade accuracy for computation by varying the number of active cores, without retraining. We use the first core token, r_1, as the `[CLS]` representation.

4 Experiments

In this section, we demonstrate that VECA learns effective global and dense visual representations without direct patch-to-patch interaction. Section 4.1 describes the training and evaluation setup; all downstream evaluations freeze the backbone and train only a linear prediction head. Section 4.2 evaluates VECA across image classification and dense prediction tasks and input resolutions. Section 4.3 ablates the architectural design choices underlying its performance and examines the compute-accuracy trade-off enabled by varying the active core budget within a single model. Section 4.4 studies the emergent semantic organization of the core tokens and its robustness across video frames.

4.1 Experimental Setup

Inspired by prior work (Heinrich et al., 2025), we train VECA and all controlled baselines by distilling from a frozen DINOv3 teacher on unlabeled Objects365 images (Shao et al., 2019), so differences isolate the attention structure rather than the training recipe. For an input image $\mathcal{I}$ and active core budget C, VECA produces a global feature $x_{\text{cls}}^{(C)}$ from the final representation of the first core token and dense features $x_{\text{dense}}^{(C)}$ from the final patch tokens. The teacher provides corresponding targets x_{cls} and x_{dense}; the remaining cores receive no teacher target. We optimize

$$\mathcal{L}(\mathcal{I}, C) = \mathcal{L}_{\text{cls}}\left(x_{\text{cls}}^{(C)}, x_{\text{cls}}\right) + \lambda \mathcal{L}_{\text{dense}}\left(x_{\text{dense}}^{(C)}, x_{\text{dense}}\right), \tag{4}$$

where $\mathcal{L}_{\text{cls}}$ aligns global representations and $\mathcal{L}_{\text{dense}}$ aligns spatial patch representations.

Training proceeds in two stages. In Stage 1, we distill VECA at 256×256 resolution while sampling active core budgets. In Stage 2, we continue training with the same objective while adapting the model to resolutions from $\{256, 384, 512, 768\}$. Additional architecture, optimization, and evaluation details are provided in the appendix.

4.2 Representation Quality and Resolution Scaling

Representation Quality. The results in Table 1 show that global and dense representations can be learned without direct patch-to-patch interaction. VECA-B/16 achieves the best performance among the evaluated linear-complexity architectures on six of seven benchmarks, spanning image

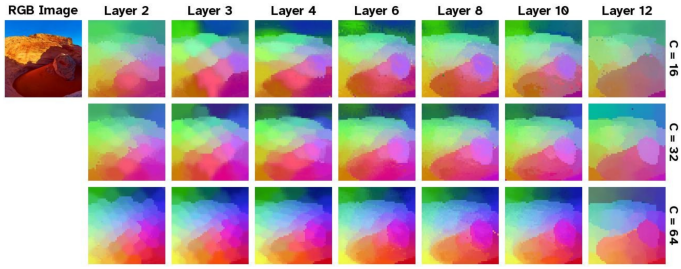

Figure 7: **Emergence of semantic core organization across depth.** Joint UMAP embeddings of the cores' attention to patches reveal increasingly coherent and semantic organization across depth, while larger core budgets yield finer semantic partitions.

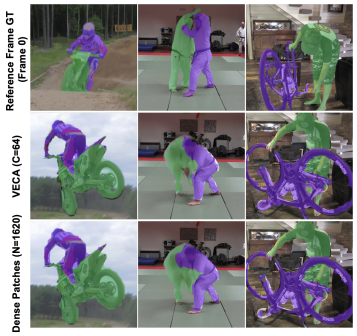

Figure 8: **Cross-frame semantic robustness using VECA cores.**

Table 4: **Quantitative evaluation of semantic core organization.** On the PartImageNet test set, many-to-one IoU (M2O) allows several cores per part, while one-to-one IoU (O2O) enforces a one-to-one matching. Colored values denote changes from Layer 1. Both metrics rise substantially with depth, showing unsupervised semantic organization; at $C = 64$, M2O grows far more than O2O, as additional cores partition parts more finely.

Budget	Metric	L1	L2	L4	L6	L8	L10	L12
$C = 8$	Many-to-one (M2O) ↑	21.62	23.88$_{+2.26}$	31.53$_{+9.91}$	45.99$_{+24.37}$	55.01$_{+33.39}$	57.51$_{+35.89}$	52.22$_{+30.60}$
	One-to-one (O2O) ↑	20.14	22.11$_{+1.97}$	25.83$_{+5.69}$	34.07$_{+13.93}$	40.86$_{+20.72}$	43.03$_{+22.89}$	38.81$_{+18.67}$
$C = 64$	Many-to-one (M2O) ↑	22.08	27.37$_{+5.29}$	41.02$_{+18.94}$	64.07$_{+41.99}$	68.17$_{+46.09}$	72.98$_{+50.90}$	72.47$_{+50.39}$
	One-to-one (O2O) ↑	17.94	21.34$_{+3.40}$	23.73$_{+5.79}$	25.11$_{+7.17}$	26.11$_{+8.17}$	26.70$_{+8.76}$	22.53$_{+4.59}$

classification, semantic segmentation, and depth estimation. VECA reaches 82.18% top-1 accuracy on ImageNet-1K (Deng et al., 2009) and 50.69 mIoU on ADE20K (Zhou et al., 2019), improving over the strongest linear-complexity baseline by 0.33 percentage points and 0.61 mIoU, respectively. It also remains close to the full-attention DINOv3-B/16 reference. At 512×512, VECA with $C = 64$ uses only 12.9% of the attention interactions of dense patch self-attention, while remaining within 0.25 mIoU of the ViT-B/16 reference on ADE20K. These results demonstrate that representations supporting both image-level recognition and dense spatial prediction can be learned without direct all-to-all patch interaction.

Scaling with Resolution. We next study the representation quality under increasing input resolutions from 256×256 to 768×768. In Figure 4, VECA improves 5.37 mIoU on ADE20K and decreases 0.0216 RMSE on NYUv2 (Silberman et al., 2012), while yielding a slight performance gain on ImageNet-1K. At 768×768, VECA outperforms other evaluated linear-complexity architectures on all four benchmarks. By contrast, Perceiver IO degrades as resolution increases, dropping from 79.49% to 77.19% top-1 accuracy on ImageNet-1K and from 80.81 to 78.40 mIoU on PASCAL VOC. These results highlight the robustness of VECA, which adapts to higher resolutions through persistent patch updates and core-to-patch interaction, whereas the fixed latent bottleneck of Perceiver IO increasingly constrains representation quality.

4.3 Design Ablations and Elastic Inference

Architectural Analysis. The architectural choices underlying VECA's scaling behavior are examined in Table 2. VECA-CoreOnly retains dense patch updates only in the first and last layers, while its intermediate layers process core tokens alone. At $C = 64$, this variant falls to 71.08% accuracy on ImageNet-1K, 40.49 mIoU on ADE20K, and 0.4584 RMSE on NYUv2. Increasing the core budget to $C = 256$ substantially improves performance, but it remains below full VECA. Notably, VECA-B with only $C = 8$ cores achieves slightly higher ImageNet-1K accuracy than VECA-CoreOnly with $C = 256$ (80.45% versus 80.30%), highlighting the importance of persistent patch updates and core-patch interaction. Dense prediction remains more sensitive to the core budget, indicating that additional cores help preserve fine spatial detail. Overall, VECA supports strong global representations even at small budgets, while larger budgets improve dense spatial reasoning. Fixed core coordinates also reduce performance, supporting VECA's learned core-coordinate design.

Elastic Computation. VECA naturally supports nested elastic training through its core structure, with essentially no degradation in full-budget performance. Table 3 shows that nested training closely matches fixed-budget training at $C = 64$ across classification, segmentation, and depth estimation.

Table 5: **Label propagation from compact reference representations.** First-frame representations propagate object labels to later frames on the DAVIS-2017 validation split, using the first frame's ground-truth annotation to label reference features. Scores are in %; Retained is $\mathcal{J}\&\mathcal{F}$ relative to dense patches. With $25\times$ fewer tokens, VECA cores retain 81% of dense performance and outperform post hoc references, whereas VECA-CoreOnly cores fail despite $4\times$ more tokens.

	Reference representation	Tokens	Compression	$\mathcal{J}$ Mean ↑	$\mathcal{F}$ Mean ↑	$\mathcal{J}\&\mathcal{F}$ Mean ↑	Retained
Full	Dense patches	1,620	$1\times$	59.3	63.2	61.2	100%
Post hoc	Uniform sampling	64	$25.3\times$	35.4	35.7	35.5	58.0%
	K-means centroids	64	$25.3\times$	44.1	45.2	44.7	73.0%
Learned cores	VECA-CoreOnly (C=256)	256	$6.3\times$	1.3	1.3	1.3	2.1%
	VECA cores (C=64)	64	$25.3\times$	**49.1**	**50.0**	**49.6**	**81.0%**

At inference time, varying the active core budget C directly adjusts the attention cost, which scales as $(2NC + C^2)$. As shown in Figure 5, performance improves smoothly with increasing core budget at 768×768, with the largest gains occurring at smaller budgets and diminishing improvements thereafter. Increasing C from 8 to 64 improves PASCAL Context by 5.27 mIoU and reduces NYUv2 by 0.0715 RMSE. At $C = 64$, VECA approaches DINOv3-B across the evaluated dense tasks, while larger budgets produce finer and more spatially localized representations. These results show that a single VECA model can trade computation for accuracy at inference time without retraining.

4.4 Emergent Core Structure and Cross-Frame Semantic Transfer

Emergent Semantic Organization. Although core tokens are introduced only as a communication interface, their repeated interaction with persistent dense patch representations leads them to develop structured and complementary visual roles without explicit supervision. Figure 6 shows that different cores respond to distinct objects, parts, and spatial regions. Figure 7 further shows that this organization emerges progressively across depth and becomes finer as the active core budget increases. On PartImageNet (He et al., 2022), many-to-one IoU increases from 21.62 to 57.51 for $C = 8$, and from 22.08 to 72.98 for $C = 64$. At the larger budget, M2O increases substantially more than O2O, indicating that additional cores specialize to finer semantic regions. Both metrics decrease slightly in the final layer, suggesting a shift toward the final training objective.

Cross-Frame Semantic Robustness. Because cores specialize to distinct objects and parts, we next test whether cores from one frame can transfer object labels to later frames. On DAVIS-2017 (Pont-Tuset et al., 2017), each of the 64 cores of the mask-annotated first frame inherits an object label from its most similar first-frame patches; each later-frame patch then retrieves its label from these labeled cores alone (details in Appendix A.13). Accurate propagation thus requires each core to represent its object consistently enough to be matched in later frames. We compare against two post hoc references of equal budget, computed from the same first-frame patch features after the forward pass and labeled identically: 64 uniformly sampled patches and 64 k-means centroids. As shown in Table 5, VECA cores reach 49.6 $\mathcal{J}\&\mathcal{F}$, retaining 81.0% of the dense baseline with $25\times$ fewer tokens and outperforming both references, including k-means (44.7). In contrast, VECA-CoreOnly reaches only 1.3 despite using 256 cores: without persistent patch updates and core-patch interaction across depth, its cores fail to retain the object structure needed for cross-frame transfer. Figure 8 shows that VECA cores recover label maps close to the dense reference.

5 Conclusion

We introduce **VECA**, a core-periphery vision architecture that learns effective global and dense representations without direct patch-to-patch interaction. VECA preserves dense patch representations while enabling global communication through a small, elastic set of learned cores, reducing attention complexity to $(2NC + C^2)$. VECA remains competitive across classification and dense prediction while supporting inference-time compute-accuracy trade-offs through the active core budget. Its continuously updated patch stream, together with core-patch interaction, allows the cores to develop semantic organization that transfers across video frames. More broadly, VECA opens opportunities for video, multi-view, and 3D vision in which cores could mediate communication across frames or views without all-to-all interaction.

AI USE STATEMENT

In this work, we used generative AI tools in a limited capacity for feedback on experimental design and minor implementation support (debugging and code review). We have not used generative AI tools to generate synthetic data, propose hypotheses, develop conceptual frameworks, formulate mathematical claims or proofs, clean or reformat datasets, translate, or interpret results, and qualitative or thematic data analysis is not applicable to this work. Additionally, and primarily, we used generative AI tools for manuscript preparation, including improving readability and formatting tables and figures. We have reviewed all AI-assisted work: LLM-edited text was revised by the authors for accuracy, table and figure edits were checked against logged results, and LLM-assisted code was validated by running and debugging the experiments. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work proposes a vision backbone architecture and does not involve human subjects or new data collection; all experiments use publicly available datasets. By reducing attention cost from quadratic to linear in the number of patches, VECA can lower the computational and energy cost of high-resolution visual inference, and its elastic core budget allows computation to be adapted to available hardware, potentially broadening access to capable vision models. Its learned cores also provide an inspectable intermediate representation that organizes images into semantically coherent regions, which may support interpretability research toward safer and more transparent AI systems.

REPRODUCIBILITY STATEMENT

We provide a detailed description of our work in the appendix, including our training data, network design choices, optimization settings, training setup, and downstream evaluation protocols. For fairness, all controlled baselines and ablations approximately match VECA's parameter count and share the same teacher (DINOv3), training data, and utilize two-stage multi-resolution training (to improve the accuracy of baseline methods on high resolution images). We also perform best-effort hyperparameter tuning for each method. In the interest of openness, we further include pseudo-code for all our network components in the supplemental. All code and trained network weights will be released under a permissive license.

REFERENCES

Marius Aasan, Odd Kolbjørnsen, Anne Schistad Solberg, and Adín Ramirez Rivera. A spitting image: Modular superpixel tokenization in vision transformers. In *European Conference on Computer Vision, MELEX Workshop*, pp. 124–142. Springer, 2024.

Marius Aasan, Martine Hjelkrem-Tan, Nico Catalano, Changkyu Choi, and Adín Ramírez Rivera. Differentiable hierarchical visual tokenization. *arXiv preprint arXiv:2511.02652*, 2025.

Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

Shantanu Acharya, Fei Jia, and Boris Ginsburg. Star attention: Efficient llm inference over long sequences. *arXiv preprint arXiv:2411.17116*, 2024.

Kwangjun Ahn, Byron Xu, Natalie Abreu, Ying Fan, Gagik Magakyan, Pratyusha Sharma, Zheng Zhan, and John Langford. Dion: Distributed orthonormalized updates. *arXiv preprint arXiv:2504.05295*, 2025.

Noah Amsel, David Persson, Christopher Musco, and Robert M Gower. The polar express: Optimal matrix sign methods and their application to the muon algorithm. *arXiv preprint arXiv:2505.16932*, 2025.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xLSTM: Extended long short-term memory. *Advances in Neural Information Processing Systems*, 37:107547–107603, 2024.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Lucas Beyer, Olivier J Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. Are we done with imagenet? *arXiv preprint arXiv:2006.07159*, 2020.

Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4009–4018, 2021.

Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *International Conference on Machine Learning*, pp. 874–883. PMLR, 2020.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022a.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, and Judy Hoffman. Hydra attention: Efficient attention with many heads. In *European conference on computer vision, CADL Workshop*, pp. 35–49. Springer, 2022b.

Guillem Brasó, Aljoša Ošep, and Laura Leal-Taixé. Native segmentation vision transformers. *arXiv preprint arXiv:2505.16993*, 2025.

Mikhail S Burtsev, Yuri Kuratov, Anton Peganov, and Grigory V Sapunov. Memory transformer. *arXiv preprint arXiv:2006.11527*, 2020.

Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1209–1218, 2018.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

Han Cai, Junyan Li, Muyan Hu, Chuang Gan, and Song Han. EfficientViT: Multi-scale linear attention for high-resolution dense prediction, 2024a. URL https://arxiv.org/abs/2205.14756.

Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. Flextron: Many-in-one flexible large language model. *arXiv preprint arXiv:2406.10260*, 2024b.

Bingyi Cao, Koert Chen, Kevis-Kokitsi Maninis, Kaifeng Chen, Arjun Karpur, Ye Xia, Sahil Dua, Tanmaya Dabral, Guangxing Han, Bohyung Han, Joshua Ainslie, Alex Bewley, Mithun Jacob, René Wagner, Washington Ramos, Krzysztof Choromanski, Mojtaba Seyedhosseini, Howard Zhou, and Andre Araujo. Tipsv2: Advancing vision-language pretraining with enhanced patch-text alignment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 29325–29335, 2026.

Lizhang Chen, Jonathan Li, Kaizhao Liang, Baiyu Su, Cong Xie, Nuo Wang Pierse, Chen Liang, Ni Lao, and Qiang Liu. Cautious weight decay. *arXiv preprint arXiv:2510.12402*, 2025a.

Yinjie Chen, Zipeng Yan, Chong Zhou, Bo Dai, and Andrew F Luo. Vision transformers with self-distilled registers. *arXiv preprint arXiv:2505.21501*, 2025b.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. *arXiv preprint arXiv:2309.16588*, 2023.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Fnu Devvrit, Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit Dhillon, Yulia Tsvetkov, Hannaneh Hajishirzi, Sham Kakade, Ali Farhadi, and Prateek Jain. MatFormer: Nested transformer for elastic inference. *Advances in Neural Information Processing Systems*, 37: 140535–140564, 2024.

Rares Dolga, Lucas Maystre, Marius Cobzarenco, and David Barber. Unifying linear-time attention via latent probabilistic modelling, 2025. URL https://arxiv.org/abs/2402.17512.

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. *arXiv preprint arXiv:2010.11929*, 2020.

Liang Duan, Charu Aggarwal, Shuai Ma, and Saket Sathe. Improving spectral clustering with deep embedding and cluster estimation. In *2019 IEEE International Conference on Data Mining (ICDM)*, pp. 170–179. IEEE, 2019.

Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111:98–136, 2015.

Alex Fang, Albin Madappally Jose, Amit Jain, Ludwig Schmidt, Alexander Toshev, and Vaishaal Shankar. Data filtering networks. *arXiv preprint arXiv:2309.17425*, 2023.

Yuxin Fang, Quan Sun, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. EVA-02: A visual representation for neon genesis. *Image and Vision Computing*, 149:105171, 2024.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Albert Gu, Ankit Gupta, Karan Goel, and Christopher Ré. On the parameterization and initialization of diagonal state space models. *Advances in neural information processing systems*, 35:35971–35983, 2022.

Naibin Gu, Zhenyu Zhang, Yuchen Feng, Yilong Chen, Peng Fu, Zheng Lin, Shuohuan Wang, Yu Sun, Hua Wu, Weiping Wang, et al. Elastic moe: Unlocking the inference-time scalability of mixture-of-experts. *arXiv preprint arXiv:2509.21892*, 2025.

Dongchen Han, Xuran Pan, Yizeng Han, Shiji Song, and Gao Huang. Flatten transformer: Vision transformer using focused linear attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5961–5971, 2023.

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 *European conference on computer vision*, pp. 124–140. Springer, 2024.

Ju He, Shuo Yang, Shaokang Yang, Adam Kortylewski, Xiaoding Yuan, Jie-Neng Chen, Shuai Liu, Cheng Yang, Qihang Yu, and Alan Yuille. Partimagenet: A large, high-quality dataset of parts. In *European Conference on Computer Vision (ECCV)*, pp. 128–145. Springer, 2022.

Greg Heinrich, Mike Ranzinger, Hongxu Yin, Yao Lu, Jan Kautz, Andrew Tao, Bryan Catanzaro, and Pavlo Molchanov. RADIOv2.5: Improved baselines for agglomerative vision foundation models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 22487–22497, 2025.

Byeongho Heo, Song Park, Dongyoon Han, and Sangdoo Yun. Rotary position embedding for vision transformer. In *European Conference on Computer Vision*, pp. 289–305. Springer, 2024.

Matthew Ho, Xiaosheng Zhao, and Benjamin D Wandelt. Ordered embeddings and intrinsic dimensionalities with information-ordered bottlenecks. *Machine Learning: Science and Technology*, 6 (3):035010, 2025.

Ali Hojjat, Janek Haberer, Soren Pirk, and Olaf Landsiedel. Thinkingvit: Matryoshka thinking vision transformer for elastic inference. *arXiv preprint arXiv:2507.10800*, 2025.

Jiaxi Hu, Yongqi Pan, Jusen Du, Disen Lan, Xiaqiang Tang, Qingsong Wen, Yuxuan Liang, and Weigao Sun. Improving bilinear RNN with closed-loop control. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Wenbo Hu, Zi-Yi Dou, Liunian H Li, Amita Kamath, Nanyun Peng, and Kai-Wei Chang. Matryoshka query transformer for large vision-language models. *Advances in Neural Information Processing Systems*, 37:50168–50188, 2024.

Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, July 2021.

Kazuki Irie, Imanol Schlag, Róbert Csordás, and Jürgen Schmidhuber. Going beyond linear transformers with recurrent fast weight programmers. *Advances in neural information processing systems*, 34:7703–7717, 2021.

Md Amirul Islam, Sen Jia, and Neil DB Bruce. How much position information do convolutional neural networks encode? *arXiv preprint arXiv:2001.08248*, 2020.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International Conference on Machine Learning*, pp. 4651–4664. PMLR, 2021.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *ICLR*, 2022.

Purvish Jajal, Nicholas John Eliopoulos, Benjamin Shiue-Hal Chou, George K Thiruvathukal, Yung-Hsiang Lu, and James C Davis. Adaperceiver: Transformers with adaptive width, depth, and tokens. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2618–2628, 2026.

Varun Jampani, Deqing Sun, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. Superpixel sampling networks. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 352–368, 2018.

Ahmadreza Jeddi, Marco Ciccone, and Babak Taati. Loopformer: Elastic-depth looped transformers for latent reasoning via shortcut modulation. *arXiv preprint arXiv:2602.11451*, 2026.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Sahil Joshi, Agniva Chowdhury, Amar Kanakamedala, Ekam Singh, Evan Tu, and Anshumali Shrivastava. Race attention: A strictly linear-time attention layer for training on outrageously large contexts. *arXiv preprint arXiv:2510.04008*, 2025.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pp. 5156–5165. PMLR, 2020.

Tsung-Wei Ke, Sangwoo Mo, and Stella X Yu. Learning hierarchical image segmentation for recognition and by recognition. *arXiv preprint arXiv:2210.00314*, 2022.

Savya Khosla, Sethuraman TV, Barnett Lee, Alexander Schwing, and Derek Hoiem. Ren: Fast and efficient region encodings from patch-based image encoders. *arXiv preprint arXiv:2505.18153*, 2025.

Dahun Kim, Anelia Angelova, and Weicheng Kuo. Region-aware pretraining for open-vocabulary object detection with vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11144–11154, 2023.

Eunwoo Kim, Chanho Ahn, and Songhwai Oh. Nestednet: Learning nested sparse structures in deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8669–8678, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7057–7075, 2020.

Shu Kong and Charless C Fowlkes. Recurrent pixel embedding for instance grouping. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9018–9028, 2018.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. Matryoshka representation learning. *Advances in Neural Information Processing Systems*, 35:30233–30249, 2022.

Saïd Ladjal, Alasdair Newson, and Chi-Hieu Pham. A pca-like autoencoder. *arXiv preprint arXiv:1904.01277*, 2019.

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*, pp. 3744–3753. PMLR, 2019.

James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. Fnet: Mixing tokens with fourier transforms. In *Proceedings of the 2022 Conference of the north American chapter of the Association for Computational Linguistics: human language technologies*, pp. 4296–4313, 2022.

Collin Leiber, Lena GM Bauer, Benjamin Schelling, Christian Böhm, and Claudia Plant. Dip-based deep embedded clustering with k-estimation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 903–913, 2021.

Collin Leiber, Niklas Strauß, Matthias Schubert, and Thomas Seidl. Dying clusters is all you need-deep clustering with an unknown number of clusters. In *2024 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 726–733. IEEE, 2024.

Kai Li and Yi Luo. Subnetwork-to-go: Elastic neural network with dynamic training and customizable inference. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6775–6779. IEEE, 2024.

Yanyu Li, Ju Hu, Yang Wen, Georgios Evangelidis, Kamyar Salahi, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Rethinking vision transformers for mobilenet size and speed. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16889–16900, 2023.

Zichong Li, Liming Liu, Chen Liang, Weizhu Chen, and Tuo Zhao. Normuon: Making muon more efficient and scalable. *arXiv preprint arXiv:2510.05491*, 2025.

James Liang, Yiming Cui, Qifan Wang, Tong Geng, Wenguan Wang, and Dongfang Liu. Clusterfomer: clustering as a universal visual learner. *Advances in neural information processing systems*, 36: 64029–64042, 2023.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*, pp. 740–755. Springer, 2014.

Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. Pay attention to mlps. *Advances in neural information processing systems*, 34:9204–9215, 2021.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025.

Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. Fastbert: a self-distilling bert with adaptive inference time. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 6035–6044, 2020.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in neural information processing systems*, 33:11525–11538, 2020.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Jiasen Lu, Christopher Clark, Sangho Lee, Zichen Zhang, Savya Khosla, Ryan Marten, Derek Hoiem, and Aniruddha Kembhavi. Unified-IO 2: Scaling autoregressive multimodal models with vision language audio and action. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26439–26455, 2024a.

Jiecheng Lu, Xu Han, Yan Sun, Viresh Pati, Yubin Kim, Siddhartha Somani, and Shihao Yang. ZeroS: Zero-sum linear attention for efficient transformers. *arXiv preprint arXiv:2602.05230*, 2026.

Zeyu Lu, Zidong Wang, Di Huang, Chengyue Wu, Xihui Liu, Wanli Ouyang, and Lei Bai. FiT: Flexible vision transformer for diffusion model. *arXiv preprint arXiv:2402.12376*, 2024b.

R Duncan Luce and Albert D Perry. A method of matrix analysis of group structure. *Psychometrika*, 14(2):95–116, 1949.

Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. *Advances in neural information processing systems*, 29, 2016.

Xuezhe Ma, Xiang Kong, Sinong Wang, Chunting Zhou, Jonathan May, Hao Ma, and Luke Zettlemoyer. Luna: Linear unified nested attention. *Advances in Neural Information Processing Systems*, 34:2441–2453, 2021.

Huanru Henry Mao. Fine-tuning pre-trained transformers into decaying fast weights. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pp. 10236–10242, 2022.

Jieru Mei, Liang-Chieh Chen, Alan Yuille, and Cihang Xie. Spformer: Enhancing vision transformer with superpixel representation. *arXiv preprint arXiv:2401.02931*, 2024.

Weikang Meng, Yadan Luo, Xin Li, Dongmei Jiang, and Zheng Zhang. Polaformer: Polarity-aware linear attention for vision transformers. *arXiv preprint arXiv:2501.15061*, 2025.

Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 891–898, 2014.

Arsha Nagrani, Shan Yang, Anurag Arnab, Aren Jansen, Cordelia Schmid, and Chen Sun. Attention bottlenecks for multimodal fusion. *Advances in neural information processing systems*, 34:14200–14213, 2021.

Tobias Christian Nauen, Sebastian Palacio, Federico Raue, and Andreas Dengel. Which transformer to favor: a comparative analysis of efficiency in vision transformers. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 6955–6966. IEEE, 2025.

Sophie Ostmeier, Brian Axelrod, Maya Varma, Michael E Moseley, Akshay Chaudhari, and Curtis Langlotz. Liere: Lie rotational positional encodings. *arXiv preprint arXiv:2406.10322*, 2024.

Namuk Park and Songkuk Kim. How do vision transformers work? *arXiv preprint arXiv:2202.06709*, 2022.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3498–3505. IEEE, 2012.

Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, Haowen Hou, Jiaju Lin, Jiaxing Liu, Janna Lu, William Merrill, et al. RWKV-7 "Goose" with expressive dynamic state evolution. *arXiv preprint arXiv:2503.14456*, 2025.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. Random feature attention. *arXiv preprint arXiv:2103.02143*, 2021.

Hao Peng, Jungo Kasai, Nikolaos Pappas, Dani Yogatama, Zhaofeng Wu, Lingpeng Kong, Roy Schwartz, and Noah A Smith. Abc: Attention with bounded-memory control. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7469–7483, 2022.

Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017.

Yifan Pu, Jixuan Ying, Qixiu Li, Tianzhu Ye, Dongchen Han, Xiaochen Wang, Ziyi Wang, Xinyu Shao, Gao Huang, and Xiu Li. Linear differential vision transformer: Learning visual contrasts via pairwise differentials. *arXiv preprint arXiv:2511.00833*, 2025.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion. *arXiv preprint arXiv:2404.07904*, 2024.

Alec Radford, Jong Wook Kim, Chris Hallacy, A. Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in neural information processing systems*, 34:12116–12128, 2021.

Mike Ranzinger, Greg Heinrich, Jan Kautz, and Pavlo Molchanov. Am-radio: Agglomerative vision foundation model reduce all domains into one. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12490–12500, 2024.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34:13937–13949, 2021.

Yongming Rao, Wenliang Zhao, Zheng Zhu, Jie Zhou, and Jiwen Lu. Gfnet: Global filter networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9): 10960–10973, 2023.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pp. 5389–5400. PMLR, 2019.

Yazhou Ren, Ni Wang, Mingxia Li, and Zenglin Xu. Deep density-based image clustering. *Knowledge-Based Systems*, 197:105841, 2020.

Oren Rippel, Michael Gelbart, and Ryan Adams. Learning ordered representations with nested dropout. In *International Conference on Machine Learning*, pp. 1746–1754. PMLR, 2014.

M Puck Rombach, Mason A Porter, James H Fowler, and Peter J Mucha. Core-periphery structure in networks. *SIAM Journal on Applied mathematics*, 74(1):167–190, 2014.

Meitar Ronen, Shahaf E Finder, and Oren Freifeld. Deepdpm: Deep clustering with an unknown number of clusters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9861–9870, 2022.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021.

Bishwajit Saha, Dmitry Krotov, Mohammed J Zaki, and Parikshit Ram. End-to-end differentiable clustering with associative memories. In *International Conference on Machine Learning*, pp. 29649–29670. PMLR, 2023.

Mert Bülent Sarıyıldız, Philippe Weinzaepfel, Thomas Lucas, Diane Larlus, and Yannis Kalantidis. UNIC: Universal classification models via multi-teacher distillation. In *European Conference on Computer Vision*, pp. 353–371. Springer, 2024.

Connor Schenck, Isaac Reid, Mithun George Jacob, Alex Bewley, Joshua Ainslie, David Rendleman, Deepali Jain, Mohit Sharma, Avinava Dubey, Ayzaan Wahid, et al. Learning the ropes: Better 2d and 3d position encodings with string. *arXiv preprint arXiv:2502.02562*, 2025.

Sohil Atul Shah and Vladlen Koltun. Deep continuous clustering. *arXiv preprint arXiv:1803.01449*, 2018.

Jinghuan Shang, Karl Schmeckpeper, Brandon B May, Maria Vittoria Minniti, Tarik Kelestemur, David Watkins, and Laura Herlant. Theia: Distilling diverse vision foundation models for robot learning. *arXiv preprint arXiv:2407.20179*, 2024.

Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li, and Jian Sun. Objects365: A large-scale, high-quality dataset for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8430–8439, 2019.

Xinwei Shen and Nicolai Meinshausen. Distributional principal autoencoders. *arXiv preprint arXiv:2404.13649*, 2024.

Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 3531–3539, January 2021.

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V 12*, pp. 746–760. Springer, 2012.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. DINOv3. *arXiv preprint arXiv:2508.10104*, 2025.

Edward W Staley and Jared Markowitz. Triangular dropout: variable network width without retraining. *arXiv preprint arXiv:2205.01235*, 2022.

Lawrence Stewart, Francis Bach, Felipe Llinares-López, and Quentin Berthet. Differentiable clustering with perturbed spanning forests. *Advances in Neural Information Processing Systems*, 36: 31158–31176, 2023.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

Hokchhay Tann, Soheil Hashemi, R Iris Bahar, and Sherief Reda. Runtime configurable deep neural networks for energy-accuracy trade-off. In *Proceedings of the eleventh IEEE/acm/ifip International Conference on Hardware/Software Codesign and System Synthesis*, pp. 1–10, 2016.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention for transformer models. In *International Conference on Machine Learning*, pp. 10183–10192. PMLR, 2021.

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 2464–2469. IEEE, 2016.

Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021.

Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, et al. Resmlp: Feedforward networks for image classification with data-efficient training. *IEEE transactions on pattern analysis and machine intelligence*, 45(4):5314–5321, 2022.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. SigLIP 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

Mojtaba Valipour, Mehdi Rezagholizadeh, Hossein Rajabzadeh, Parsa Kavehzadeh, Marzieh Tahaei, Boxing Chen, and Ali Ghodsi. Sortednet: A scalable and generalized framework for training modular deep neural networks. *arXiv preprint arXiv:2309.00255*, 2023.

Chase van de Geijn, Timo Lüddecke, Polina Turishcheva, and Alexander S Ecker. A circular argument: Does rope need to be equivariant for vision? *arXiv preprint arXiv:2511.08368*, 2025.

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

Yaoxiang Wang, Qingguo Hu, Yucheng Ding, Ruizhe Wang, Yeyun Gong, Jian Jiao, Yelong Shen, Peng Cheng, and Jinsong Su. Training matryoshka mixture-of-experts for elastic inference-time expert utilization. *arXiv preprint arXiv:2509.26520*, 2025.

Yixuan Wei, Han Hu, Zhenda Xie, Zheng Zhang, Yue Cao, Jianmin Bao, Dong Chen, and Baining Guo. Contrastive learning rivals masked image modeling in fine-tuning via feature distillation. *arXiv preprint arXiv:2205.14141*, 2022.

Haixu Wu, Jialong Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Flowformer: Linearizing transformers with conservation flows. In *International Conference on Machine Learning*, pp. 24226–24242, 2022.

Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8817–8826, 2018.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning*, pp. 478–487. PMLR, 2016.

Zhonglin Xie, Yiman Fong, Haoran Yuan, and Zaiwen Wen. Accelerating optimization via differentiable stopping time. *arXiv preprint arXiv:2505.22509*, 2025.

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 *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 14138–14148, 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 18134–18144, 2022.

Fuzhao Xue, Valerii Likhosherstov, Anurag Arnab, Neil Houlsby, Mostafa Dehghani, and Yang You. Adaptive computation with elastic input sequence. In *International Conference on Machine Learning*, pp. 38971–38988. PMLR, 2023.

Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *International Conference on Machine Learning*, pp. 3861–3870. PMLR, 2017.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

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*, pp. 10809–10818, 2022.

Haoran You, Yunyang Xiong, Xiaoliang Dai, Bichen Wu, Peizhao Zhang, Haoqi Fan, Peter Vajda, and Yingyan Celine Lin. Castling-ViT: Compressing self-attention via switching towards linear-angular attention at vision transformer inference. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14431–14442, 2023.

Hao Yu, Tangyu Jiang, Shuning Jia, Shannan Yan, Shunning Liu, Haolong Qian, Guanghao Li, Shuting Dong, and Chun Yuan. Comrope: Scalable and robust rotary position embedding parameterized by trainable commuting angle matrices. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 4508–4517, 2025.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

Wang Zeng, Sheng Jin, Lumin Xu, Wentao Liu, Chen Qian, Wanli Ouyang, Ping Luo, and Xiaogang Wang. Tcformer: Visual recognition via token clustering transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9521–9535, 2024.

Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Ré. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. *arXiv preprint arXiv:2402.04347*, 2024a.

Xiao Zhang, Travis Martin, and Mark EJ Newman. Identification of core-periphery structure in networks. *Physical Review E*, 91(3):032803, 2015.

Yitian Zhang, Huseyin Coskun, Xu Ma, Huan Wang, Ke Ma, Xi Chen, Derek H Hu, and Yun Fu. Slicing vision transformer for flexible inference. *Advances in Neural Information Processing Systems*, 37:42649–42671, 2024b.

Yu Zhang, Songlin Yang, Ruijie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, et al. Gated slot attention for efficient linear-time sequence modeling. *Advances in Neural Information Processing Systems*, 37:116870–116898, 2024c.

Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127:302–321, 2019.

Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI magazine*, 17(3):73–83, 1996.

A TECHNICAL APPENDICES AND SUPPLEMENTARY MATERIAL

Sections

1. Architecture details of VECA (Section A.1)
2. Training and optimization details (Section A.2)
3. Core-Periphery attention pseudocode (Section A.3)
4. Baseline model details (Section A.4)
5. Implementation details for downstream tasks (Section A.5)
6. Additional multi-resolution evaluation across tasks (Section A.6)
7. Additional core-to-patch similarity visualizations (Section A.7)
8. Additional classification results (Section A.8)
9. Additional emergent behavior visualizations (Section A.9)
10. Efficiency analysis and computational cost (Section A.10)
11. Additional patch-to-patch similarity visualizations (Section A.11)
12. Additional architectural ablations and elastic computation studies (Section A.12)
13. Cross-frame semantic transferability (Section A.13)

A.1 ARCHITECTURE DETAILS OF VECA

This section specifies the concrete VECA model configurations used in our experiments. The main paper defines the core-mediated attention mechanism and its motivation; here we report implementation-level architectural settings.

Model configurations. We instantiate VECA at four model sizes: Small, Small+, Base, and Large. All variants use a 16×16 patch embedding, a maximum core capacity $M = 64$, and nested active-core budgets $\{8, 16, 24, 32, 40, 48, 56, 64\}$. Effectively, we implement a granularity of 8 cores for nested training. The model scale is controlled by the number of layers, hidden dimension, number of attention heads, and feed-forward expansion ratio.

Table S.1: **VECA architecture configurations.** All variants use patch size 16×16, maximum core capacity $M = 64$, and active-core budgets $\{8, 16, 24, 32, 40, 48, 56, 64\}$.

Model	Layers	Hidden dim.	Heads	MLP ratio
VECA-Small	12	384	6	2.67
VECA-Small+	12	384	6	4.00
VECA-Base	12	768	12	2.67
VECA-Large	24	1024	16	2.67

Parameter counts. VECA closely matches the parameter count of the corresponding DINOv3 backbone at each model scale. The small increase comes from the learnable core tokens, core coordinates, and coordinate-update layers.

Table S.2: **Parameter counts.** We compare VECA with the corresponding DINOv3 backbone at each model size.

Model	VECA	DINOv3
Small	21.63 M	21.59 M
Small+	28.72 M	28.68 M
Base	85.73 M	85.64 M
Large	303.20 M	303.08 M

Shared architectural settings. Table S.3 summarizes the settings shared across model sizes. Attention is implemented using PyTorch scaled dot-product attention.

Table S.3: **Shared architectural settings across VECA variants.**

Component	Setting
Patch size	16×16
Input channels	3
Patch embedding	Strided convolution
Patch embedding normalization	None
Dropout	0.0
Maximum core capacity	$M = 64$
Core-token chunk size	8
Active-core budgets	$\{8, 16, 24, 32, 40, 48, 56, 64\}$
Global representation	First core token r_1
Dense representation	All patch tokens Z
FFN type	SwiGLU
Attention implementation	PyTorch scaled dot-product attention (SDPA)

Token and parameter layout. The learnable core-token bank is denoted by

$$R_M = \{r_1, \ldots, r_M\},$$

where M is the maximum core capacity. The bank is stored as chunked parameters, with 8 core tokens per chunk and 8 chunks in total. For an active budget C, the model activates the first $C/8$ chunks, yielding the active prefix

$$R_C = R_M[:C] = (r_1, \ldots, r_C).$$

Given patch tokens $Z = \{z_1, \ldots, z_N\}$, the input sequence to each block is ordered as

$$X = [R_C; Z].$$

This fixed ordering is used by the attention implementation in Section A.3.

Transformer block. Each block uses pre-normalization, core-mediated attention, and a SwiGLU feed-forward network:

$$X'_\ell = X_\ell + \mathrm{VECA_Attn}(\mathrm{LN}(X_\ell)), \qquad X_{\ell+1} = X'_\ell + \mathrm{FFN}(\mathrm{LN}(X'_\ell)).$$

The exact implementation is given in Section A.3.

Feed-forward network. The feed-forward network uses a gated SwiGLU form:

$$[u, v] = W_1 z, \qquad \mathrm{FFN}(z) = W_2 (\mathrm{SiLU}(u) \odot v).$$

This design is used for all model sizes; VECA-Small+ differs from VECA-Small only by using a larger MLP expansion ratio.

Core coordinates and positional encoding. Each patch token is assigned a fixed two-dimensional coordinate from the patch grid. Each core token is assigned a learnable two-dimensional coordinate initialized by farthest-point sampling in the normalized image plane. For core i at layer ℓ, we denote the unconstrained coordinate by $\rho_i^\ell \in \mathbb{R}^2$ and the bounded coordinate used for RoPE by $u_i^\ell \in [-1, 1]^2$:

$$u_i^\ell = \tanh(\rho_i^\ell).$$

Both patch and core tokens use two-dimensional rotary positional encoding. For layers $\ell > 0$, the core coordinate state is updated using a learned residual:

$$\Delta\rho_i^\ell = W_\ell^{\mathrm{pos}} r_i^\ell, \qquad \rho_i^\ell = \rho_i^{\ell-1} + \alpha_\ell \Delta\rho_i^\ell, \qquad u_i^\ell = \tanh(\rho_i^\ell),$$

where r_i^ℓ is the feature of core token i at layer ℓ, W_ℓ^{pos} is a lightweight coordinate-update head, and α_ℓ is a learned scalar initialized to a small value.

Output representations. After the final transformer block, LayerNorm is applied to the full token sequence. The first final core token is used as the global image representation,

$$y(x) = r_1^L,$$

and the final patch tokens are used as the dense representation,

$$Z^L(x) = \{z_1^L, \ldots, z_N^L\}.$$

Recent models have adopted randomly augmented embeddings (Kim et al., 2023; Ranzinger et al., 2024), 2D axial RoPE (Fang et al., 2024; Lu et al., 2024a;b; Siméoni et al., 2025), or more sophisticated (sometimes non-commutative) 2D RoPE variants (Heo et al., 2024; Ostmeier et al., 2024; Schenck et al., 2025; van de Geijn et al., 2025; Yu et al., 2025). For this work, we follow DINOv3 with the adoption of 2D axial RoPE, and leave the exploration of other positional encodings to future work.

A.2 TRAINING AND OPTIMIZATION DETAILS

This section provides the optimization and training settings used for VECA. We focus on implementation details that are not specified in the main paper.

Training stages. VECA is trained in two stages. First, we train the model at 256×256 resolution. Second, we continue from the single-resolution checkpoint and perform multi-resolution finetuning for $50{,}000$ steps over resolutions $\{256, 384, 512, 768\}$. Both stages use the same frozen-teacher objective and the same nested active-core training scheme.

Table S.4: **Training stages.**

Stage	Resolution	Duration
Single-resolution training	256×256	135 epochs
Multi-resolution finetuning	$\{256, 384, 512, 768\}$	50,000 steps

Training objective. For an image x and sampled active-core budget C, the student produces a global representation $y^{(C)}(x) \in \mathbb{R}^D$ and dense patch representation $Z^{(C)}(x) = \{z_i^{(C)}(x)\}_{i=1}^N$, where $z_i^{(C)}(x) \in \mathbb{R}^D$. The frozen teacher provides corresponding targets $y^\star(x)$ and $Z^\star(x) = \{z_i^\star(x)\}_{i=1}^N$. We optimize

$$\mathcal{L}(x, C) = \mathcal{L}_{\text{global}}\Big(y^{(C)}(x), y^\star(x)\Big) + \lambda_{\text{dense}}\mathcal{L}_{\text{dense}}\Big(Z^{(C)}(x), Z^\star(x)\Big).$$

The global loss is cosine distance:

$$\mathcal{L}_{\text{global}} = 1 - \frac{\big\langle y^{(C)}(x), y^\star(x)\big\rangle}{\big\|y^{(C)}(x)\big\|_2 \, \|y^\star(x)\|_2}.$$

The dense loss combines patch-wise cosine distance and mean-squared error:

$$\mathcal{L}_{\text{dense}} = \frac{1}{N}\sum_{i=1}^{N}\left(1 - \frac{\big\langle z_i^{(C)}(x), z_i^\star(x)\big\rangle}{\big\|z_i^{(C)}(x)\big\|_2 \, \|z_i^\star(x)\|_2}\right) + \beta_{\text{mse}}\frac{1}{ND}\sum_{i=1}^{N}\Big\|z_i^{(C)}(x) - z_i^\star(x)\Big\|_2^2.$$

We set $\lambda_{\text{dense}} = 1.0$, $\beta_{\text{mse}} = 1.0$, and use normalization epsilon 10^{-6}.

Training data. VECA is trained on Objects365 images. For each training image, we apply both the global representation loss and the dense patch-level loss using targets from the frozen teacher. No labeled class supervision is used during backbone training.

Teacher models. Each VECA model is trained with the corresponding DINOv3 teacher at the same scale. The teacher is frozen throughout training. When the teacher includes register tokens, we remove them before constructing dense patch targets.

Table S.5: **Teacher models used for VECA training.** All teachers are DINOv3 models from the `facebook/*-pretrain-lvd1689m` family.

Student	Teacher	Teacher micro-batch
VECA-Small	ViT-S/16	128
VECA-Small+	ViT-S+/16	128
VECA-Base	ViT-B/16	128
VECA-Large	ViT-L/16	64

Batch size. For VECA-Small, VECA-Small+, and VECA-Base, we use a per-rank batch size of 128 during single-resolution training. For VECA-Large, we use a per-rank micro-batch size of 64 and accumulate gradients for 2 steps.

Table S.6: **Per-rank batch settings for single-resolution training.**

Model	Micro-batch	Grad. accum.	Effective batch
VECA-Small	128	1	128
VECA-Small+	128	1	128
VECA-Base	128	1	128
VECA-Large	64	2	128

Input preprocessing and augmentation. For single-resolution training, images are resized to a short side of 288 pixels and cropped to 256×256. For multi-resolution finetuning, the crop resolution is sampled from $\{256, 384, 512, 768\}$, and the short side is resized to 1.125 times the crop resolution. Training uses random crop, random horizontal flip, and `TrivialAugmentWide`. Validation uses center crop. All images are normalized with

$$\mu = (0.485, 0.456, 0.406), \qquad \sigma = (0.229, 0.224, 0.225).$$

Active-core budget sampling. During both stages, one active-core budget is sampled at each optimizer step from

$$C \in \{8, 16, 24, 32, 40, 48, 56, 64\}$$

with sampling weights

$$(1, 1, 2, 2, 3, 3, 4, 4).$$

The sampled budget is broadcast to all distributed ranks before the forward pass. Validation uses the full active-core budget unless otherwise specified.

Optimization. We use a two-optimizer setup. Selected two-dimensional linear weights are optimized with NorMuon (Jordan et al., 2024; Li et al., 2025), while the remaining trainable parameters are optimized with AdamW (Kingma & Ba, 2014; Loshchilov & Hutter, 2017). We utilize the Dion library (Ahn et al., 2025) for all optimizers. The AdamW group includes core tokens, core coordinates, coordinate-update layers, positional scaling parameters, biases, normalization parameters, patch embedding parameters, and selected output projection parameters. All other parameters are optimized with NorMuon.

Table S.7: **Optimizer and schedule settings.**

Setting	Single-resolution	Multi-resolution finetuning
AdamW learning rate	4×10^{-4}	8×10^{-5}
NorMuon learning rate	4×10^{-4}	8×10^{-5}
Minimum learning rate	5×10^{-5}	3×10^{-5}
Weight decay	0.015	0.01
AdamW epsilon	10^{-8}	10^{-8}
Schedule	Warmup + cosine	Warmup + cosine
Mixed precision	bfloat16	bfloat16
Training hardware	6 NVIDIA L40S GPUs	6 NVIDIA L40S GPUs

NorMuon settings. For NorMuon, we use momentum $\mu = 0.95$, second-moment coefficient $\beta_2 = 0.95$, Nesterov updates, RMS-normalized AdamW compatible learning-rate adjustment (Liu et al., 2025), and cautious weight decay (Chen et al., 2025a). The orthogonalization routine uses five Polar Express (Amsel et al., 2025) iterations. We do not use parameter flattening.

Distributed training and checkpointing. All models are trained using PyTorch DistributedDataParallel on 6 NVIDIA L40S GPUs. The active-core budget is sampled on the main process and broadcast to all ranks before each optimizer step. We use bfloat16 automatic mixed precision for both student and teacher forward passes. Validation metrics are averaged across distributed ranks. During multi-resolution finetuning, validation is performed over the resolution and budget grid, and the best checkpoint is selected according to the averaged validation loss.

Table S.8: **NorMuon settings.**

Setting	Value
Momentum μ	0.95
β_2	0.95
Nesterov	True
Learning-rate adjustment	RMS norm
Cautious weight decay	True
Newton–Schulz / Polar Express iterations	5
Triton kernels	False
Parameter flattening	False

A.3 CORE-MEDIATED ATTENTION PSEUDOCODE

This section gives implementation-level code snippets for the VECA attention block. The model configurations are provided in Section A.1. We use M for the maximum core capacity, $C \leq M$ for the active number of core tokens, N for the number of patch tokens, and D for the hidden dimension. The ordered core-token bank is

$$R_M = (r_1, \ldots, r_M).$$

For an active budget C, the model uses the active prefix

$$R_C = R_M[:C] = (r_1, \ldots, r_C) \in \mathbb{R}^{C \times D}.$$

Given patch tokens $Z \in \mathbb{R}^{N \times D}$, the input sequence is ordered as

$$X = [R_C; Z] \in \mathbb{R}^{(C+N) \times D}.$$

The listings below are generalized from the training implementation and omit model-size-specific constants. In the Python code, the variable `active_c` corresponds to the mathematical budget C.

Core-mediated attention. The core-mediated attention layer uses separate query, key, and value projections. After applying two-dimensional RoPE to queries and keys, active core tokens attend to the full sequence, while patch tokens attend only to the active core-token prefix.

```
class CoreMediatedAttention(nn.Module):
    def __init__(self, dim: int, num_heads: int, dropout: float = 0.0):
        super().__init__()
        assert dim % num_heads == 0
        self.num_heads = num_heads
        self.head_dim = dim // num_heads

        self.q_proj = nn.Linear(dim, dim, bias=True)
        self.k_proj = nn.Linear(dim, dim, bias=True)
        self.v_proj = nn.Linear(dim, dim, bias=True)
        self.out_proj = nn.Linear(dim, dim, bias=True)

        self.drop_p = float(dropout)
        self.out_drop = nn.Identity() if dropout == 0.0 else nn.Dropout(dropout)

    def forward(self, x: torch.Tensor, rope: Rotary2D,
                coords: torch.Tensor, active_c: int) -> torch.Tensor:
        B, T, D = x.shape
        H, Hd = self.num_heads, self.head_dim
        assert 1 <= active_c < T

        q = self.q_proj(x)
        k = self.k_proj(x)
        v = self.v_proj(x)

        q = q.view(B, T, H, Hd).transpose(1, 2)
        k = k.view(B, T, H, Hd).transpose(1, 2)
        v = v.view(B, T, H, Hd).transpose(1, 2)

        cos, sin = rope.cos_sin(coords)
        q = rope.apply(q, cos.to(q.dtype), sin.to(q.dtype))
        k = rope.apply(k, cos.to(k.dtype), sin.to(k.dtype))

        p = self.drop_p if self.training else 0.0

        # Active core tokens R_C attend to all active cores and all patches.
        y_core = F.scaled_dot_product_attention(
            q[:, :, :active_c, :], k, v,
            dropout_p=p,
            is_causal=False,
        )

        # Patch tokens Z attend only to the active core-token prefix R_C.
        y_patch = F.scaled_dot_product_attention(
            q[:, :, active_c:, :],
            k[:, :, :active_c, :],
            v[:, :, :active_c, :],
            dropout_p=p,
            is_causal=False,
        )

        y = torch.cat([y_core, y_patch], dim=2)
        y = y.transpose(1, 2).contiguous().view(B, T, D)
        y = self.out_proj(y)
        return self.out_drop(y)
```

For visualization, the corresponding attention pattern can be written with output rows ordered as (Z, R) and source columns ordered as (R, Z):

$$A = \begin{bmatrix} A_{Z\leftarrow R} & 0 \\ A_{R\leftarrow R} & A_{R\leftarrow Z} \end{bmatrix}.$$

The zero block denotes the removed patch-to-patch interaction $A_{Z\leftarrow Z}$. This display ordering is only for illustrating the sparsity pattern; the implementation keeps the token sequence ordered as $[R_C; Z]$.

VECA transformer block. Each block applies pre-normalized core-mediated attention followed by a pre-normalized SwiGLU feed-forward network.

```
class VECABlock(nn.Module):
    def __init__(self, dim: int, num_heads: int,
                 mlp_ratio: float, dropout: float = 0.0):
        super().__init__()
        self.norm_attn = nn.LayerNorm(dim)
        self.attn = CoreMediatedAttention(dim, num_heads, dropout)

        hidden = int(dim * mlp_ratio)
        self.norm_ffn = nn.LayerNorm(dim)
        self.fc1 = nn.Linear(dim, 2 * hidden, bias=True)
        self.fc2 = nn.Linear(hidden, dim, bias=True)
        self.drop = nn.Identity() if dropout == 0.0 else nn.Dropout(dropout)

    def forward(self, x: torch.Tensor, rope: Rotary2D,
                coords: torch.Tensor, active_c: int) -> torch.Tensor:
        x = x + self.attn(
            self.norm_attn(x),
            rope=rope,
            coords=coords,
            active_c=active_c,
        )

        y = self.norm_ffn(x)
        u, v = self.fc1(y).chunk(2, dim=-1)
        y = F.silu(u) * v
        y = self.drop(y)
        y = self.fc2(y)
        y = self.drop(y)

        return x + y
```

Active core selection and coordinate update. VECA stores the core-token bank R_M and the corresponding core-coordinate bank in chunks. For a budget C, the forward pass activates the first C core tokens and updates only their coordinates layer by layer.

```
class VECAEncoder(nn.Module):
    def _build_active_cores(self, batch_size: int, active_c: int):
        n_chunks = active_c // self.core_chunk_size

        cores = torch.cat(
            [self.core_token_chunks[j] for j in range(n_chunks)],
            dim=1,
        ).expand(batch_size, -1, -1)

        core_raw = torch.cat(
            [self.core_coord_chunks[j] for j in range(n_chunks)],
            dim=1,
        ).expand(batch_size, -1, -1)

        return cores, core_raw

    @torch.no_grad()
    def _patch_grid(self, hp: int, wp: int,
                    device: torch.device) -> torch.Tensor:
        y = (torch.arange(hp, device=device).float() + 0.5) / hp * 2.0 - 1.0
        x = (torch.arange(wp, device=device).float() + 0.5) / wp * 2.0 - 1.0
        gy, gx = torch.meshgrid(y, x, indexing="ij")
        return torch.stack([gx, gy], dim=-1).reshape(-1, 2)
```

VECA forward pass. The full encoder first computes patch tokens Z, concatenates the active core-token prefix R_C with the patch sequence, and then applies VECA blocks. The first final core token is returned as the global representation, and the final patch tokens are returned as dense features.

```python
class VECAEncoder(nn.Module):
    def forward(self, images: torch.Tensor,
                active_c: Optional[int] = None):
        B = images.shape[0]
        active_c = self.num_core_tokens if active_c is None else int(active_c)

        assert 1 <= active_c <= self.num_core_tokens
        assert active_c % self.core_chunk_size == 0

        patches, (hp, wp) = self.patch_embed(images)
        cores, core_raw = self._build_active_cores(B, active_c)

        patch_coords = self._patch_grid(hp, wp, patches.device)
        patch_coords = patch_coords.unsqueeze(0).expand(B, -1, -1)
        patch_coords = self.rope.augment_coords_patch(patch_coords)

        core_coords = torch.tanh(core_raw)
        x = torch.cat([cores, patches], dim=1)

        for layer_idx, block in enumerate(self.blocks):
            if layer_idx > 0:
                active_cores = x[:, :active_c, :]
                delta = self.pos_linears[layer_idx - 1](active_cores)
                alpha = self.pos_alpha[layer_idx - 1].to(delta.dtype)
                core_raw = core_raw + alpha * delta
                core_coords = torch.tanh(core_raw)

            coords = torch.cat([core_coords, patch_coords], dim=1)
            x = block(x, self.rope, coords, active_c)

        x = self.final_norm(x)

        cores = x[:, :active_c, :]
        patches = x[:, active_c:, :]

        global_feature = cores[:, 0, :]
        dense_features = patches
        return global_feature, dense_features
```

A.4 BASELINE MODEL DETAILS

This section summarizes the usage and implementation of baseline backbones and controlled linear-complexity alternatives.

We perform rigorous controlled experiments in this work. All controlled linear-complexity alternatives utilize approximately the same number of parameters as our model, is trained on the same data-diet, utilizes the same strong DINOv3 teacher, and undergoes multi-resolution training to improve their accuracy at high resolutions. We also perform best-effort hyperparmeter tuning to improve their accuracy. We show conclusively via these carefully controlled experiments that our method yields results stronger than alternative techniques.

A.4.1 VISION BACKBONES

Baseline backbones. We compare VECA with a set of widely used vision backbones spanning contrastive vision-language pretraining, self-supervised representation learning, and recent vision foundation models. All baseline models are evaluated using their publicly released checkpoints.

Table S.9: **Baseline models used in our evaluation.** The asterisk indicates a resolution-specific checkpoint variant.

Family	Backbone	Checkpoint
CLIP	ViT-B/16	`openai/clip-vit-base-patch16`
OpenCLIP	ViT-B/16	`ViT-B-16/laion2b_s34b_b88k`
DFN-CLIP	ViT-B/16	`apple/DFN2B-CLIP-ViT-B-16`
DINOv2	ViT-B/14	`facebook/dinov2-base`
DINOv2-Reg	ViT-B/14	`facebook/dinov2-with-registers-base`
DINOv3	ViT-B/16	`facebook/dinov3-vitb16`
SigLIP2*	ViT-B/16	`google/siglip2-base-patch16`
AM-RADIOv2.5 (C-RADIOv3)	ViT-B/16	`nvidia/C-RADIOv3-B`

Model usage. For each baseline, we use the official or checkpoint-compatible loading interface associated with the released model. All backbone parameters are kept frozen during downstream evaluation. For OpenCLIP and DFN-CLIP, we use the visual backbone from the released checkpoint. For AM-RADIOv2.5 (C-RADIOv3) (Heinrich et al., 2025), we use the model-provided external preprocessor together with the released backbone.

A.4.2 CONTROLLED ARCHITECTURE BASELINES

To isolate the effect of the token-interaction architecture, we additionally construct a quadratic ViT reference and several linear-complexity baselines. All controlled models use a 16×16 patch embedding, a 768-dimensional representation, and an approximately 86M-parameter budget. Their parameters are learned through the same distillation and multiresolution training protocol used for VECA. Feed-forward dimensions are adjusted where necessary to match the parameter budget.

Every controlled backbone produces both a global image representation and a dense patch representation. We therefore apply the same frozen-backbone evaluation protocol to all architectures.

Quadratic softmax reference. ViT-B/16 (Dosovitskiy et al., 2020) is a standard 12-layer transformer with 12-head softmax self-attention, interpolated learned positional embeddings, and SwiGLU feed-forward layers. Unlike the linear-complexity models below, it forms a complete token-to-token attention matrix and serves as the parameter-matched quadratic reference.

Set Transformer. Set Transformer (Lee et al., 2019) uses six Induced Set Attention Blocks (ISABs), each with 64 block-specific learned inducing tokens. Inducing tokens first summarize all patch tokens, after which patches read back from this summary, preserving dense outputs without direct all-to-all patch attention. To further strengthen this baseline, we add fixed 2D Fourier coordinates through a learned projection; this is a vision adaptation because the original method treats its input as an unordered set. A single learned pooling seed produces the global representation.

Perceiver IO. Perceiver IO (Jaegle et al., 2022) maps patches into 256 learned latents, processes them with 12 self-attention blocks, and decodes them through cross-attention. Fixed 2D Fourier coordinates with learned projections encode both input patches and dense output queries, consistent with the original Perceiver design; the latent tokens themselves have no spatial positions. One learned query produces the global representation, while spatial queries produce dense patch features.

Efficient Attention. Efficient Attention (Shen et al., 2021) replaces softmax self-attention with a dual-softmax factorization. Queries are normalized across feature channels and keys are normalized across tokens, allowing the multiplication order to be rearranged so that an $N \times N$ attention matrix is never constructed. We use this operator in all 12 transformer blocks while preserving the standard CLS and patch-token sequence.

Flowformer. Flowformer (Wu et al., 2022) uses noncausal Flow Attention in all 12 blocks. It applies sigmoid feature maps to queries and keys, then uses conservation-based normalization, source competition, and sink allocation to control how information is distributed among tokens. The resulting operation has linear complexity in sequence length.

Hydra Attention. Hydra Attention (Bolya et al., 2022b) normalizes queries and keys along the feature dimension and summarizes the sequence through a globally aggregated feature-wise context. Each token interacts with this shared context rather than with every other token individually, yielding linear complexity in the number of tokens. We replace attention in all 12 transformer blocks with this operator.

PolaFormer. PolaFormer (Meng et al., 2025) separates positive and negative query-key interactions so that both similar and opposite-polarity relations can contribute to the output. It additionally uses learned power transformations and output gating. A 5×5 depthwise-convolution branch provides local spatial information that is otherwise difficult to preserve in factorized linear attention.

Nyströmformer. Nyströmformer (Xiong et al., 2021) approximates softmax attention using 64 landmarks formed by averaging contiguous token groups. Interactions between the full token sequence and these landmarks approximate the complete attention matrix. We use six Newton-Schulz iterations to approximate the landmark-matrix pseudoinverse and include the method's depthwise residual convolution.

Advantages of VECA over alternative linear-time approaches. Our extensive controlled experiments demonstrate that VECA exceeds prior linear-time baselines on the majority of evaluated image classification and dense tasks, while exhibiting excellent high-resolution dense feature quality and stability. The VECA-CoreOnly experiment shows that our improved accuracy cannot be solely explained by architectural hyperparameter decisions, such as choice of activation function or Pre-Norm attention blocks. Our ablations support the importance of repeated bidirectional patch-core interaction and persistent refinement of both token streams as two reasons for the improved performance: **First**, is that we enable bidirectional communication between the dense patches and the cores in every single layer. In contrast, in Perceiver and Perceiver IO, information flows from input tokens to latent tokens, but not from latent tokens back to input tokens; **Second**, is the refinement and processing of the dense tokens and core tokens in every layer, whereas prior Perceiver and Set Transformer approaches refine either only the latent tokens (Perceiver family) or only the input tokens (Each ISAB block in Set Transformer has its own learned inducing tokens, and does not carry the preceding block's latent-token state forward). By enabling attention based bidirectional communication between patch tokens and core tokens in every layer, and progressively refining both sets of tokens, our work demonstrates that VECA achieves highly competitive performance compared to full attention approaches. We further introduce core count as an additional axis for elastic inference in VECA. This work suggests that VECA introduces promising directions for future exploration.

A.5 Implementation details for downstream tasks

This section provides implementation details for downstream evaluation. We separate the classification protocol from dense prediction protocols.

A.5.1 Image Classification

Linear probing protocol. For image classification benchmarks, we freeze the visual backbone and train only a linear classification head on top of the extracted global feature. For VECA, the global feature is the first final core token. For each baseline, we use the corresponding global visual representation defined by its released implementation. The backbone is kept in evaluation mode throughout linear probing.

Optimization. The linear head is trained with SGD using momentum 0.9, learning rate 0.05, and weight decay 0. We train for $12{,}500$ steps with batch size 128 and a cosine learning-rate schedule. Validation is performed every $1{,}250$ steps, and we keep the checkpoint with the best validation top-1 accuracy.

Resolution and preprocessing. For classification benchmarks, all backbones with patch size 16 are evaluated with a 256×256 crop, and those with patch size 14 (DINOv2, DINOv2-Reg) with a 252×252 crop. Before cropping, we resize the short side to 1.125 times the crop size. We use the checkpoint-compatible normalization for each backbone, and interpolate absolute positional embeddings to the input resolution where needed.

Evaluation metric. We report top-1 accuracy for classification benchmarks.

A.5.2 Dense Prediction

Linear probing protocol. For semantic segmentation and depth estimation benchmarks, we train a linear prediction head while keeping the backbone frozen. The entire evaluation pipeline is implemented based on the MMSegmentation framework (Contributors, 2020). For both VECA and all baselines, we extract dense patch features and append the **CLS** token to each patch representation to enrich the feature representations (Bhat et al., 2021). For SigLIP 2 (Tschannen et al., 2025), which does not include a **CLS** token, we directly use patch features for depth evaluation. All backbones remain in evaluation mode throughout linear probing.

Optimization. The linear head is trained using the AdamW optimizer with betas $(0.9, 0.999)$, a learning rate of 0.01, and a weight decay of 0.0001, together with a polynomial learning rate scheduler. The number of training steps depends on the benchmark. We use $20{,}000$ steps for PASCAL VOC 2012 (VOC), $40{,}000$ for PASCAL Context-60 (Context) (Mottaghi et al., 2014) and ADE20K-150 (ADE), $80{,}000$ for COCO-Stuff (Stuff) (Caesar et al., 2018) and COCO-Object (Object) (Lin et al., 2014), and $38{,}400$ for NYUv2 and KITTI (Geiger et al., 2013). The batch size is set to 8. For evaluation, we retain the checkpoint with the best validation mIoU or RMSE, depending on the task.

Resolution and preprocessing. For dense prediction benchmarks, each backbone is evaluated using a fixed crop resolution of 512×512 for models with patch size 16 and 518×518 for models with patch size 14. Before cropping, we resize the shorter side to 512 or 518 and apply random cropping during training. For CLIP, OpenCLIP (Ilharco et al., 2021), and DFN-CLIP (Fang et al., 2023) models with absolute positional embeddings, we interpolate the positional embeddings to match the input resolution.

Evaluation metrics. We report mIoU for semantic segmentation benchmarks and RMSE for depth estimation benchmarks.

A.6 Additional multi-resolution evaluation across tasks

This section provides additional multi-resolution evaluation against state-of-the-art vision backbones. We consider image classification, dense prediction, and qualitative comparisons.

A.6.1 Image Classification

Protocol. We evaluate VECA, DINOv3, and DINOv2 across model sizes and input resolutions. VECA and DINOv3 use resolutions $\{256, 384, 512, 768\}$, while DINOv2 uses the nearest patch-aligned resolutions $\{252, 392, 518, 770\}$ for its patch size of 14. All backbones are frozen, and a separate linear head is trained for each model and resolution following Section A.5.1.

Table S.10: **Multi-resolution classification.** Each cell reports ImageNet-1K top-1 accuracy. Headers show the VECA/DINOv3 resolution, with the corresponding DINOv2 resolution in parentheses.

Model	256 (252)	384 (392)	512 (518)	768 (770)
DINOv3-S	77.31	78.43	78.81	78.55
DINOv3-S+	79.95	80.77	81.23	81.19
DINOv3-B	83.56	84.14	84.15	84.08
DINOv3-L	86.40	86.67	86.67	86.74
DINOv2-S	78.77	79.76	79.94	79.41
DINOv2-B	82.45	83.02	83.34	82.81
DINOv2-L	84.58	84.79	84.74	84.41
VECA-S	74.63	75.39	75.74	75.42
VECA-S+	77.03	77.58	77.82	77.50
VECA-B	82.18	82.92	82.89	82.87
VECA-L	84.51	84.67	84.73	84.53

Summary. Table S.10 shows that VECA remains competitive with DINOv2 and DINOv3 across resolutions. At Base scale, VECA-B retains $98.4\%-98.6\%$ of DINOv3-B ImageNet-1K accuracy. At Large scale, VECA-L retains $97.5\%-97.8\%$ on ImageNet-1K.

A.6.2 Dense Prediction

Protocol. We compare VECA and DINOv3 on multi-resolution semantic segmentation and monocular depth estimation, as DINOv3 provides the strongest dense prediction baseline. Both are evaluated at resolutions $\{256, 384, 512, 768\}$. All backbones are frozen, and a separate linear head is trained for each model and resolution following Section A.5.2.

Table S.11: **Multi-resolution semantic segmentation.** Each cell reports mIoU on (Context, ADE, Stuff, Object).

Model	256	384	512	768
DINOv3-S	51.78 / 41.28 / 41.74 / 54.60	54.06 / 44.55 / 43.82 / 57.83	55.01 / 46.01 / 44.67 / 59.49	55.10 / 46.62 / 45.04 / 59.83
DINOv3-S+	51.81 / 41.67 / 42.13 / 55.54	54.87 / 46.44 / 44.95 / 59.51	55.95 / 48.57 / 45.86 / 61.12	56.38 / 49.53 / 46.27 / 61.69
DINOv3-B	55.02 / 46.61 / 45.87 / 60.67	57.07 / 49.53 / 47.97 / 63.67	57.74 / 51.35 / 48.41 / 64.65	58.40 / 52.37 / 49.04 / 65.78
DINOv3-L	56.50 / 49.75 / 47.36 / 62.55	58.33 / 52.66 / 48.93 / 65.13	59.13 / 54.56 / 49.71 / 66.53	59.70 / 55.52 / 50.12 / 67.28
VECA-S	50.98 / 40.61 / 40.49 / 52.91	52.85 / 43.19 / 42.86 / 55.92	53.46 / 44.82 / 43.45 / 57.46	53.87 / 45.28 / 43.77 / 57.96
VECA-S+	51.19 / 41.07 / 41.32 / 54.16	54.02 / 44.78 / 43.53 / 57.50	54.68 / 46.55 / 44.33 / 58.76	55.02 / 47.03 / 44.78 / 59.08
VECA-B	54.91 / 46.09 / 45.43 / 59.71	56.79 / 49.24 / 47.27 / 62.69	57.46 / 50.69 / 47.92 / 62.58	58.05 / 51.46 / 48.37 / 64.14
VECA-L	56.66 / 50.05 / 47.36 / 62.28	58.61 / 52.99 / 48.92 / 64.94	59.64 / 54.17 / 49.53 / 66.19	60.23 / 55.54 / 50.06 / 66.85

Summary. Tables S.11 and S.12 show that VECA remains competitive with DINOv3 across resolutions and model scales. Both models generally benefit from higher resolution, while VECA scales linearly with the number of input tokens compared with quadratic self-attention in DINOv3. At Base scale, VECA-B remains close to DINOv3-B across the evaluated tasks. At Large scale, VECA-L matches or exceeds DINOv3-L on several benchmarks, including Context, ADE, and NYUv2. These results highlight VECA's favorable accuracy-computation trade-off at high resolutions.

Table S.12: **Multi-resolution depth estimation.** Each cell reports RMSE on $(\mathrm{NYUv2}, \mathrm{KITTI})$.

Model	256	384	512	768
DINOv3-S	0.4565 / 3.2402	0.4252 / 3.1130	0.4169 / 2.9083	0.4206 / 2.8943
DINOv3-S+	0.4651 / 3.3499	0.4225 / 2.9987	0.4143 / 2.8127	0.4140 / 2.8125
DINOv3-B	0.3897 / 2.9636	0.3727 / 2.8962	0.3684 / 2.6915	0.3665 / 2.6840
DINOv3-L	0.3663 / 2.9207	0.3460 / 2.7836	0.3446 / 2.5346	0.3445 / 2.4821
VECA-S	0.4528 / 3.2776	0.4360 / 3.2106	0.4347 / 3.0130	0.4298 / 3.0289
VECA-S+	0.4597 / 3.3009	0.4316 / 3.1318	0.4252 / 2.9878	0.4231 / 3.0176
VECA-B	0.3920 / 2.9765	0.3768 / 2.8976	0.3705 / 2.7252	0.3704 / 2.7468
VECA-L	0.3546 / 2.8792	0.3409 / 2.8019	0.3379 / 2.5904	0.3382 / 2.5634

A.6.3 Additional Qualitative Comparison

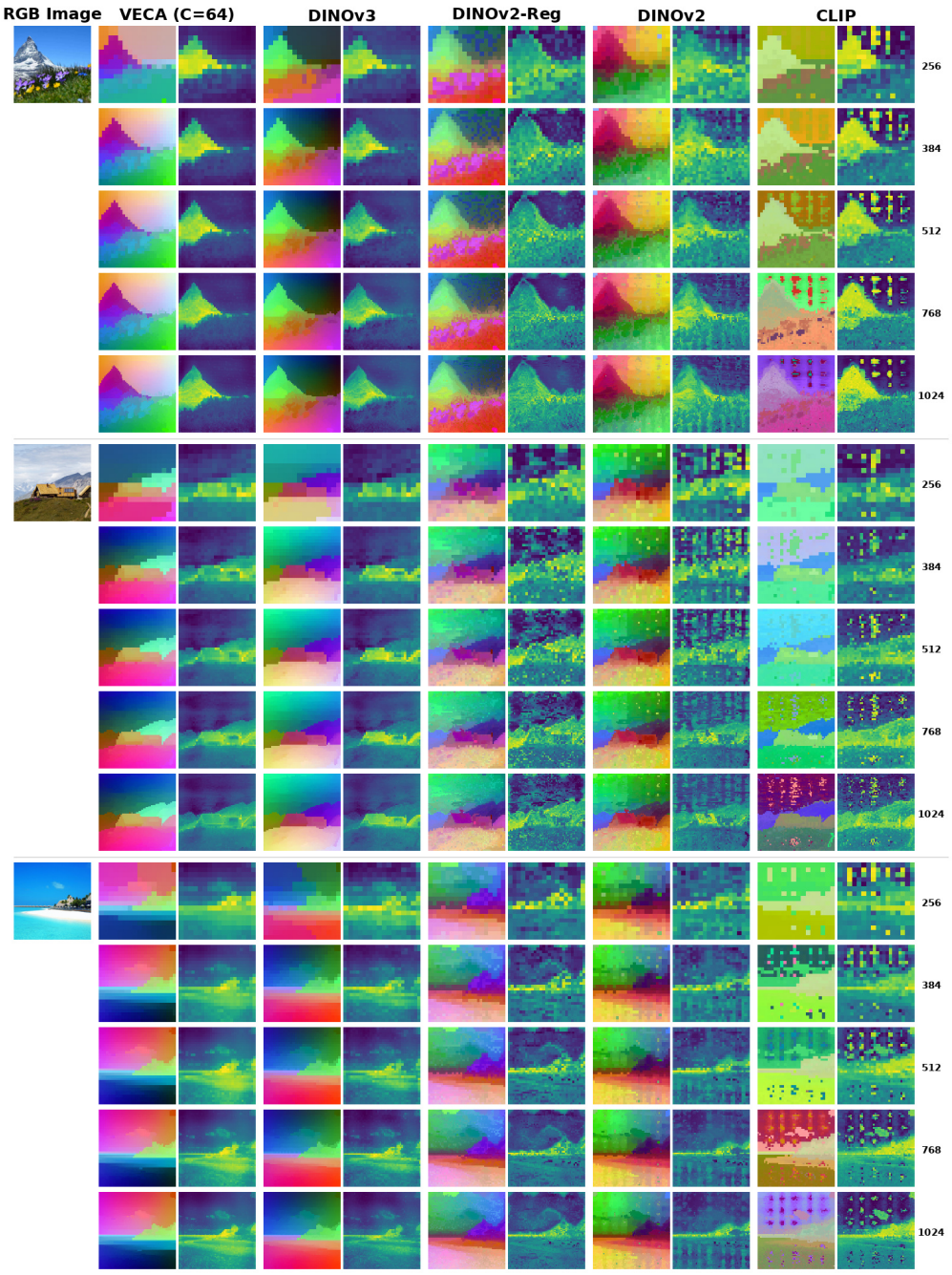

Figure S.1: **Additional qualitative multi-resolution comparison.** We compare dense patch representations from VECA and baseline ViTs across resolutions. VECA preserves coherent object boundaries and smooth semantic regions, while several baselines show more fragmented similarity patterns at higher resolutions.

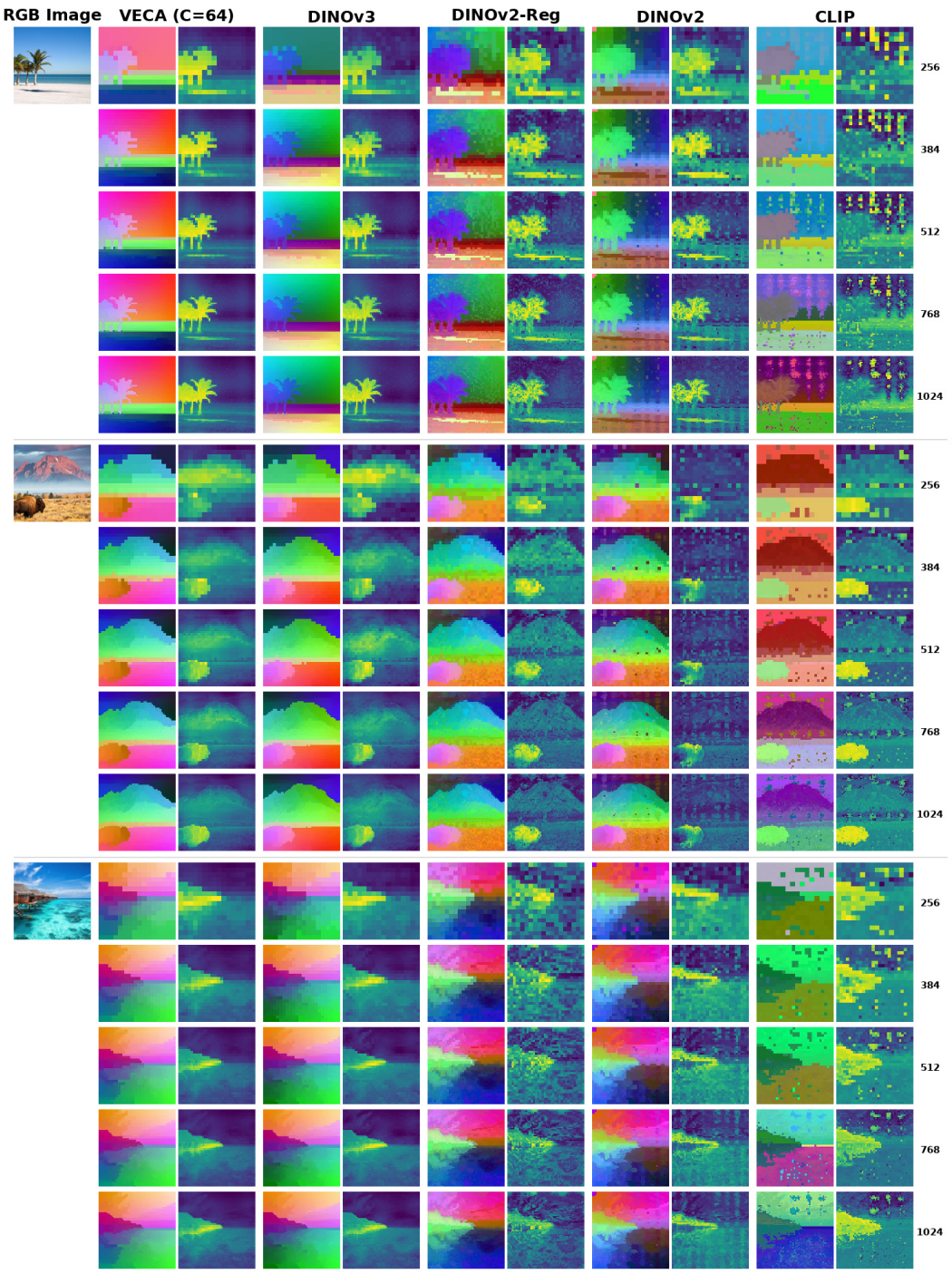

Figure S.2: **Additional qualitative multi-resolution comparison.** We compare dense patch representations from VECA and baseline ViTs across resolutions. VECA preserves coherent object boundaries and smooth semantic regions, while several baselines show more fragmented similarity patterns at higher resolutions.

A.7 ADDITIONAL CORE-TO-PATCH SIMILARITY VISUALIZATIONS

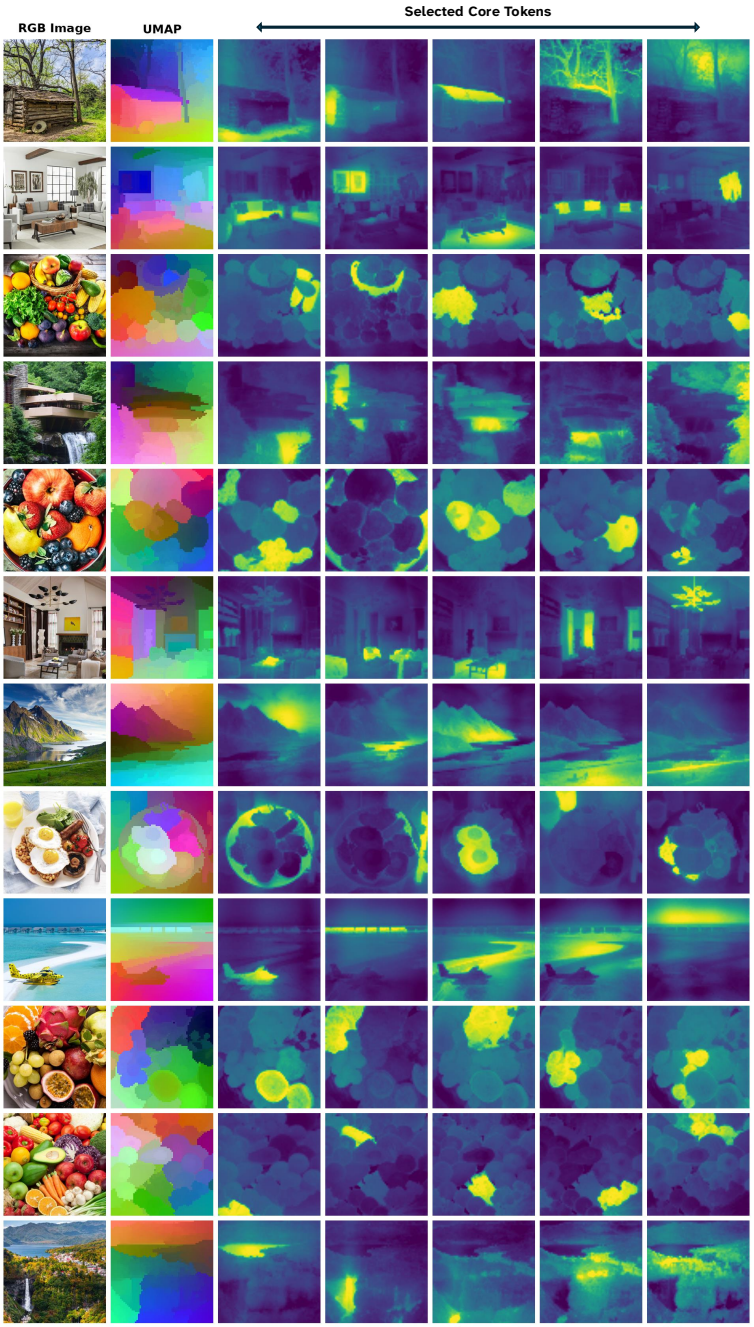

Figure S.3: **Additional core-to-patch similarity visualizations.** For each RGB image, we show a UMAP projection of dense patch features followed by similarity maps for selected core tokens from our model. Warmer colors indicate image regions with higher cosine similarity to a given core token. The selected cores attend to complementary objects, parts, and spatial regions, illustrating that different core tokens specialize in distinct visual structures while preserving coherent scene-level organization.

A.8 ADDITIONAL CLASSIFICATION RESULTS

We follow the frozen-backbone linear-probing protocol in Section A.5.1. ImageNetV2 (Recht et al., 2019) and ImageNet-ReaL (Beyer et al., 2020) reuse the ImageNet-1K linear head without adaptation, while each remaining dataset uses an independently trained linear head. The evaluation used 256 for patch-16 models and 252 for patch-14 models.

Table S.13: **Additional frozen-backbone classification results.** We report top-1 accuracy. Bold indicates the best result among the linear-complexity controlled baselines and VECA variants.

Model	IN1K	V2	ReaL	Places	Food	Aircraft	Pets	Flowers
External pretrained references								
DINOv3-B/16	83.56	74.92	88.65	55.39	94.47	81.97	96.54	99.69
DINOv2-Reg-B/14	83.44	74.75	88.18	54.81	92.87	70.78	95.97	99.67
DINOv2-B/14	82.45	72.98	86.84	53.37	92.02	74.02	95.61	99.59
SigLIP2-B/16-256	81.95	73.04	87.69	56.18	94.70	62.80	92.89	97.98
DFN2B-CLIP-B/16	81.06	71.35	86.56	55.79	93.32	66.49	93.59	98.70
AM-RADIOv2.5-B/16	80.53	70.60	85.78	54.13	91.64	52.18	93.54	98.52
CLIP-B/16	79.33	69.31	84.93	55.27	92.42	56.65	92.72	96.36
OpenCLIP-B/16	79.18	69.05	84.72	55.35	91.33	62.89	91.91	96.50
Controlled linear-complexity baselines								
Set Transformer	80.15	70.53	86.74	55.50	90.42	67.90	95.58	98.42
Perceiver IO	79.49	69.29	86.23	55.30	89.22	66.91	94.55	97.56
Efficient Attention	81.24	71.80	87.49	55.66	91.54	71.50	95.56	99.04
FlowFormer	81.11	71.16	87.36	55.53	91.40	71.47	95.34	98.88
Hydra Attention	79.12	68.42	85.96	55.42	89.15	67.33	95.07	98.00
PolaFormer	81.85	72.10	87.77	55.73	92.40	**74.59**	**95.94**	99.32
Nyströmformer	81.34	71.96	87.54	55.73	91.82	73.33	95.50	99.15
Ours								
VECA-B/16 ($C = 64$)	**82.18**	**72.25**	**87.87**	**55.82**	**92.44**	74.47	95.67	**99.35**
VECA-B/16 ($C = 8$)	80.45	69.18	86.45	55.30	90.04	70.72	95.12	98.98

Summary. VECA-B/16 outperforms the controlled linear-complexity baselines on six of eight benchmarks, while PolaFormer is slightly stronger on Aircraft and Pets (Parkhi et al., 2012). At $C = 8$, VECA retains competitive transfer performance, showing that its inference budget can be reduced substantially while preserving useful global representations.

A.9 Additional emergent behavior visualizations

In this section, we provide additional visualizations of how VECA organizes patch representations through its learned core-token interface.

A.9.1 Analysis Method

To visualize how the cores organize the image, we look at how each core attends to the patches. At layer ℓ, we take core c as the query and the patches as keys. Instead of raw attention, we use value-weighted attention (Kobayashi et al., 2020): the norm of the attention-weighted value vector after the output projection, summed over heads and normalized over all keys. This gives a weight s^{ℓ}_{cp} for each core c and patch p.

Each patch is then described by a C-dimensional feature, one entry per active core, which gives a map of shape $H/P \times W/P \times C$. We embed these features into three dimensions with UMAP, jointly across the selected layers, and display the result as an $H/P \times W/P \times 3$ RGB image. We omit the first layer, where attention mostly reflects initialization.

A.9.2 ADDITIONAL VISUALIZATIONS

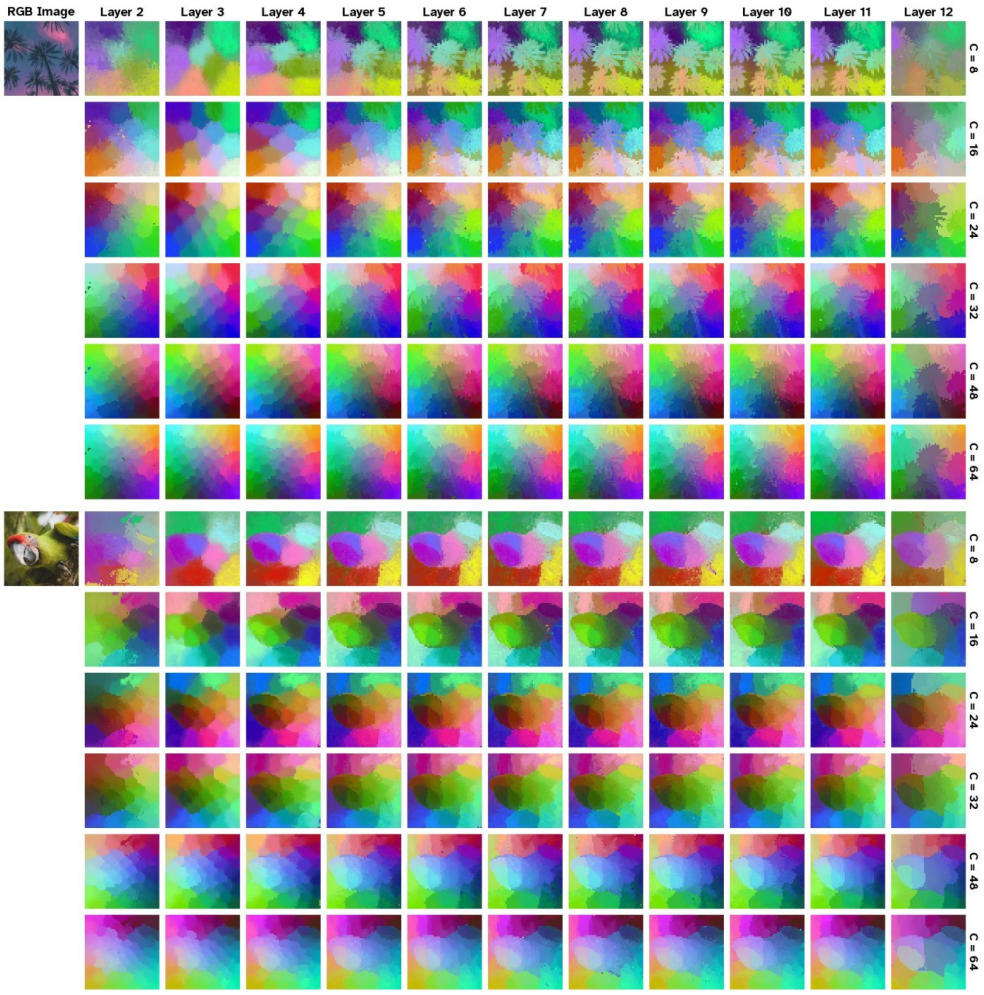

Figure S.4: **Additional layer-wise emergent behavior visualizations.** We visualize UMAP embeddings of patch contribution profiles across active core queries over layers and core budgets. Please zoom in for details.

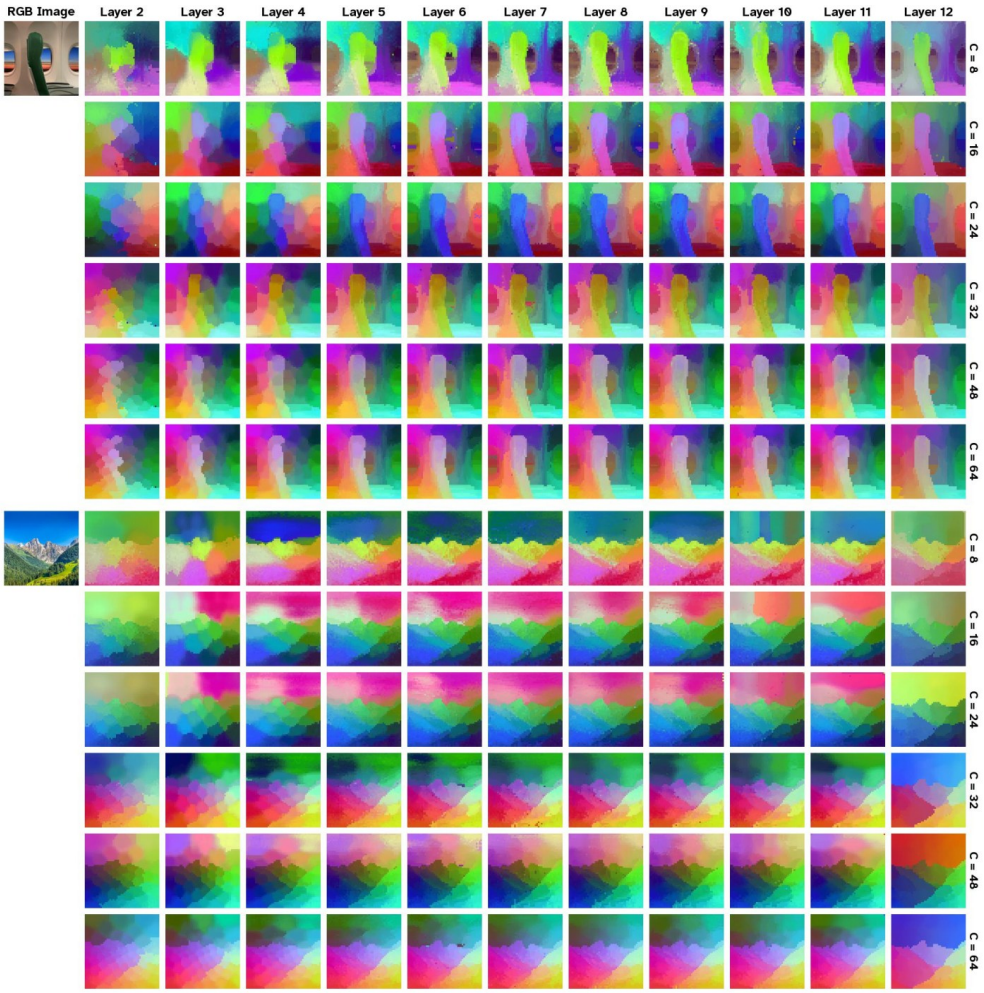

Figure S.5: **Additional layer-wise emergent behavior visualizations.** We visualize UMAP embeddings of patch contribution profiles across active core queries over layers and core budgets. Please zoom in for details.

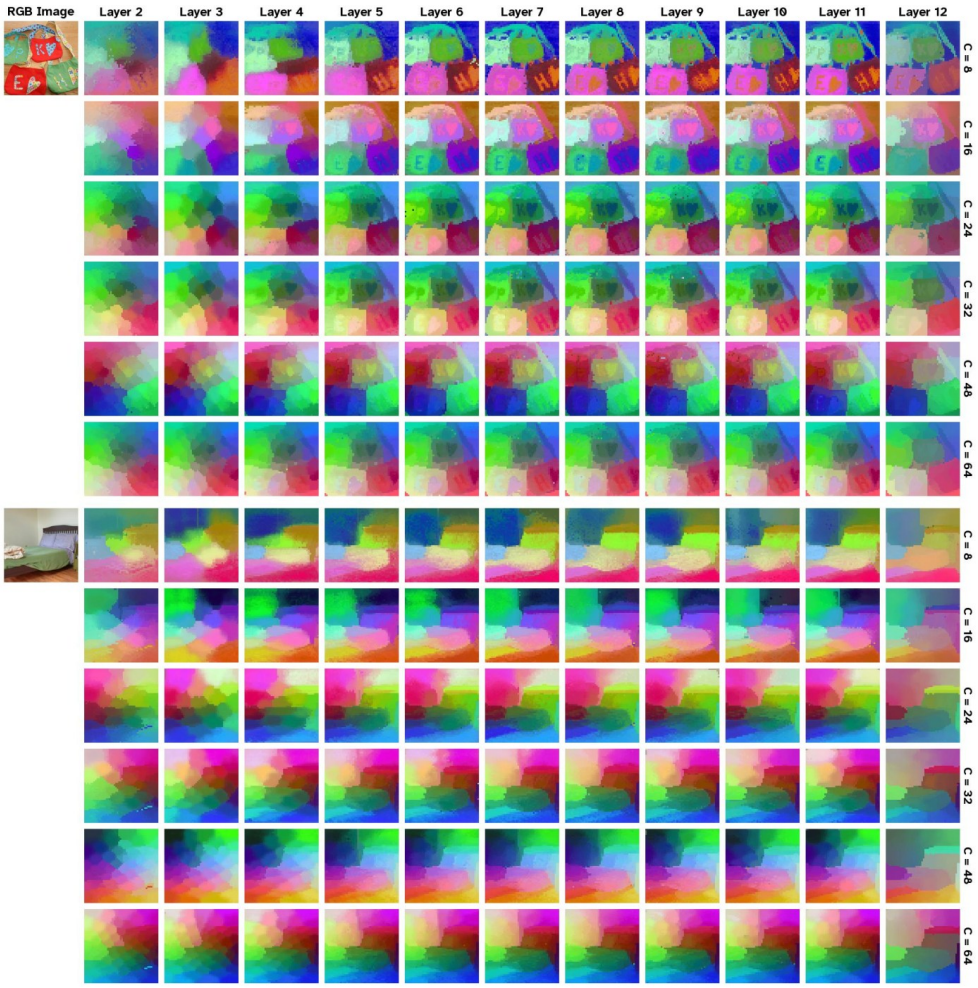

Figure S.6: **Additional layer-wise emergent behavior visualizations.** We visualize UMAP embeddings of patch contribution profiles across active core queries over layers and core budgets. Please zoom in for details.

A.10 Efficiency analysis and computational cost

This section provides efficiency analysis for VECA. The main computational advantage of VECA comes from replacing dense patch-to-patch self-attention with core-mediated attention. For an active core budget C and N patch tokens, the attention interaction cost scales as

$$N^2 \quad \longrightarrow \quad 2NC + C^2.$$

Thus, when $C \ll N$, the benefit becomes more pronounced as the input resolution increases.

Benchmark design. We compare VECA with the corresponding DINOv3 backbone at the Small, Base, and Large scales. To isolate the cost of the attention mechanism, we benchmark the same projection-attention-projection path for both models: query, key, and value projections, scaled dot-product attention, and the output projection. For VECA, patch tokens attend only to the active core tokens, while core tokens attend to the full sequence. For DINOv3, all tokens use standard full self-attention. Unless otherwise stated, VECA uses the full active core budget $C = 64$. We measure FLOPs over a resolution sweep from 256 to 1024, reported per image for a single attention layer and counting one multiply-add as one FLOP, as in Table 2; GPU latency is measured on an NVIDIA L40S 48GB GPU with bfloat16 precision.

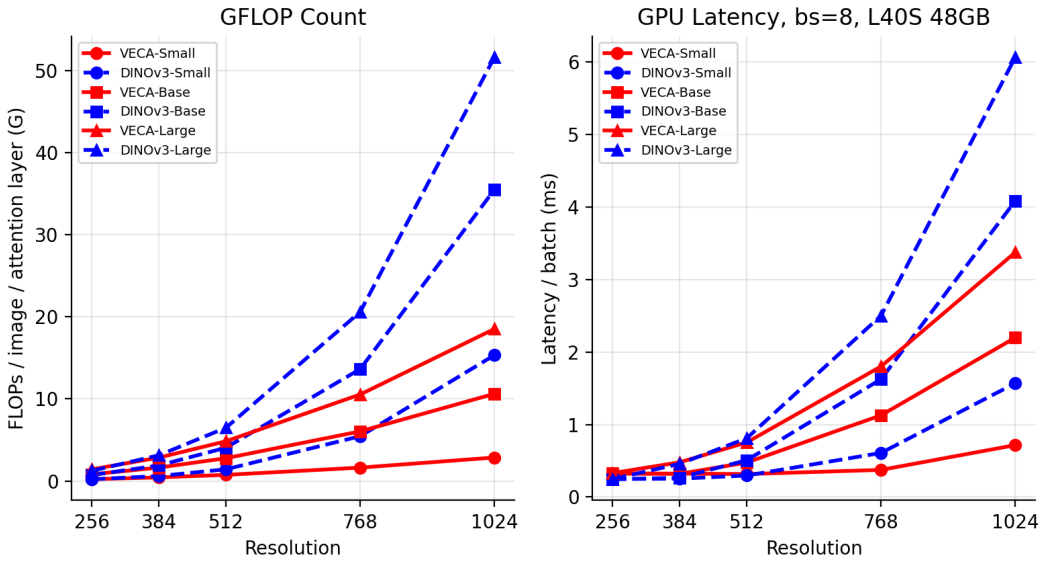

Figure S.7: **Resolution sweep for attention cost.** We compare VECA and DINOv3 across input resolutions for Small, Base, and Large models. Left: FLOPs per image for a single attention layer. Right: GPU latency at batch size 8 on an NVIDIA L40S 48GB GPU. VECA becomes increasingly efficient as resolution grows because the removed patch-to-patch attention term dominates at high token counts.

Resolution scaling. Figure S.7 shows that the efficiency advantage grows with resolution. At 256, the C added core tokens make VECA slightly more expensive, since the sequence is short and projection/kernel overheads dominate; VECA requires fewer FLOPs from 384 onward and has lower latency from 512 (Base, Large) or 768 (Small). At the largest evaluated resolution, VECA substantially reduces both FLOPs and GPU latency across all model sizes. For VECA-Small, FLOPs decrease from 15.34G to 2.86G, a 5.36× reduction, while latency decreases from 1.57ms to 0.72ms. For VECA-Base, FLOPs decrease from 35.51G to 10.63G, and latency decreases from 4.08ms to 2.20ms. For VECA-Large, FLOPs decrease from 51.64G to 18.53G, and latency decreases from 6.07ms to 3.37ms.

Batch-size scaling. We further evaluate GPU latency of the attention residual block over batch sizes from 1 to 128 at 1024×1024 resolution. This tests whether the advantage persists under larger batched workloads.

Table S.14: **Attention cost at the largest evaluated resolution** (1024×1024)**.** FLOPs are reported per image for a single attention layer. GPU latency is reported per batch with batch size 8 on an NVIDIA L40S 48GB GPU.

	FLOPs (G)			Latency (ms)		
Model	VECA	DINOv3	Red.	VECA	DINOv3	Speedup
Small	2.86	15.34	5.36×	0.72	1.57	2.19×
Base	10.63	35.51	3.34×	2.20	4.08	1.86×
Large	18.53	51.64	2.79×	3.37	6.07	1.80×

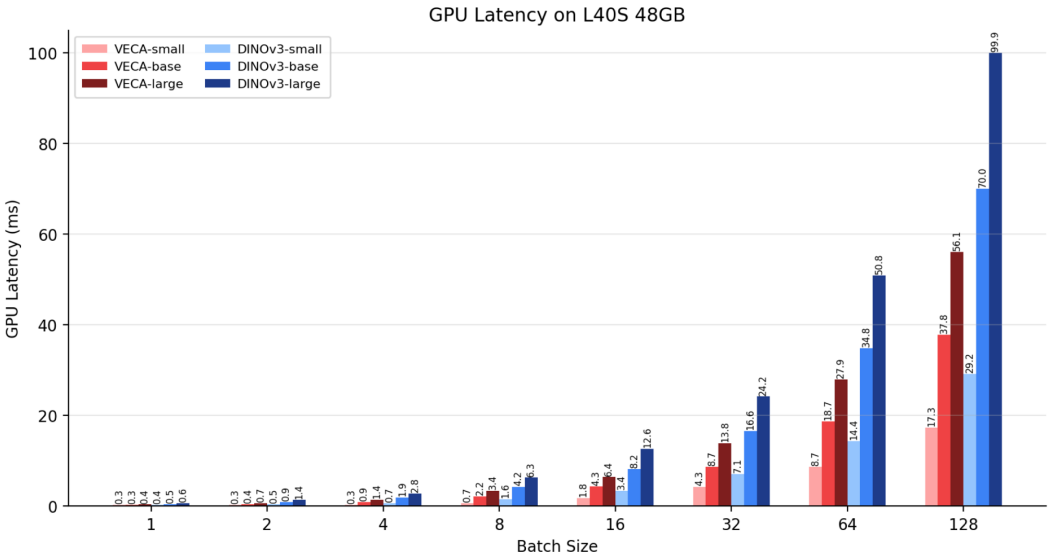

Figure S.8: **Batch-size scaling at high resolution.** GPU latency is measured on an NVIDIA L40S 48GB GPU. The VECA attention residual block remains consistently faster than the corresponding DINOv3 residual block across model sizes and batch sizes.

As shown in Figure S.8, the VECA attention block maintains lower latency across the batch-size sweep. At the largest batch size, VECA-Small reduces latency from 29.18ms to 17.33ms, VECA-Base reduces latency from 70.04ms to 37.79ms, and VECA-Large reduces latency from 99.92ms to 56.07ms. These correspond to $1.68\times$, $1.85\times$, and $1.78\times$ speedups, respectively.

Summary. Overall, the efficiency results support the intended scaling behavior of VECA. The gains are most visible at high resolution, where standard ViT self-attention becomes dominated by patch-to-patch interactions. For a fixed active core budget C, VECA reduces this cost while retaining bidirectional communication between the compact core set and the dense patch sequence, yielding lower FLOPs and GPU latency once the patch count is large enough for the patch-to-patch term to dominate.

A.11 ADDITIONAL PATCH-TO-PATCH SIMILARITY VISUALIZATIONS

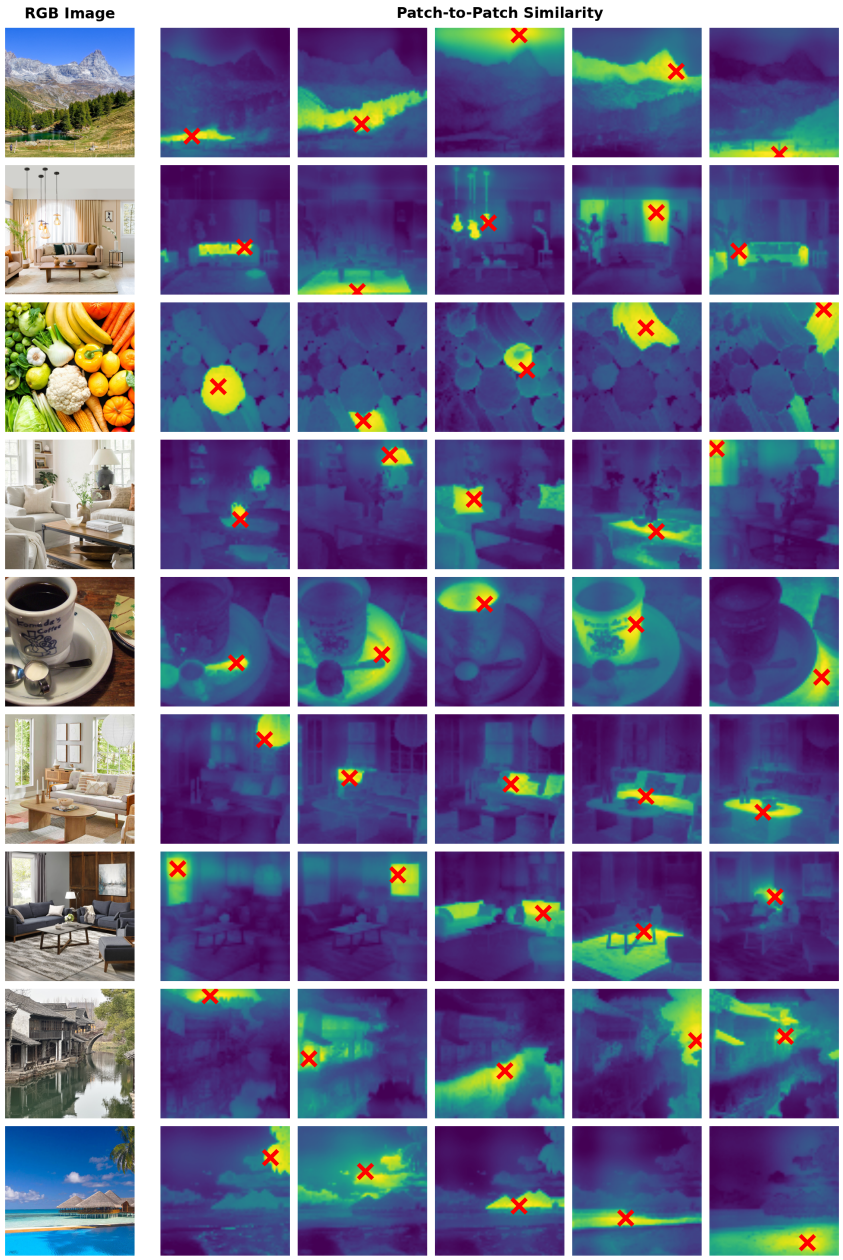

Figure S.9: **Patch-to-patch similarity visualizations.** For each RGB image, we visualize patch-to-patch cosine similarity maps from our model. The red cross marks the selected reference patch, and warmer colors indicate patches with higher feature similarity to that reference location. Across diverse scenes, the similarity maps often recover semantically or structurally related regions, suggesting that the learned patch representations preserve coherent object-, part-, and layout-level relationships.

A.12 Additional architectural ablations and elastic computation studies

We present additional ablations of VECA's architectural design, followed by a further analysis of its elastic computation behavior when the number of active core tokens is varied for classification and dense prediction.

A.12.1 Ablation Studies

Design. Set Transformer's ISAB block (Lee et al., 2019) also routes patch communication through learned tokens, but creates fresh tokens at every layer and uses two separate cross-attention steps. VECA's cores instead persist across depth and share a single attention operator with the patches. VECA-LocalCore keeps VECA's operator but resets its cores at every layer, so Set Transformer $\rightarrow$ VECA-LocalCore isolates the operator and VECA-LocalCore $\rightarrow$ VECA-B isolates core persistence. We also distill VECA-B from a TIPSv2 teacher (Cao et al., 2026) to validate our choice of DINOv3.

Table S.15: **Additional ablations of VECA's design.** Set Transformer and VECA-LocalCore both use layer-local latent tokens; VECA-LocalCore shares VECA's attention operator but resets its cores at every layer. Colored values denote differences from VECA-B at $C = 64$.

Variant	IN1K Acc. ↑	ADE mIoU ↑	NYUv2 RMSE ↓
Layer-local latent tokens			
Set Transformer (inducing tokens = 64) (Lee et al., 2019)	$80.15_{-2.03}$	$48.34_{-2.35}$	$0.3958_{+0.0253}$
VECA-LocalCore ($C = 64$)	$80.59_{-1.59}$	$49.32_{-1.37}$	$0.3819_{+0.0114}$
Alternative teacher			
VECA-B, TIPSv2 teacher ($C = 64$)	$74.69_{-7.49}$	$48.81_{-1.88}$	$0.3971_{+0.0266}$
Full VECA architecture			
VECA-B ($C = 64$)	**82.18**	**50.69**	**0.3705**

Results. Both factors contribute, with core persistence providing the larger gain on IN1K and ADE20K: making the cores persistent (VECA-LocalCore $\rightarrow$ VECA-B) improves them by $+1.59$ and $+1.37$, compared with $+0.44$ and $+0.98$ from switching to VECA's operator. Distilling from TIPSv2 instead of DINOv3 lowers IN1K accuracy by 7.49 points, supporting DINOv3 as the stronger teacher for our setting.

A.12.2 Elastic Computation: Classification

Protocol. We vary the active core-token budget $C \in \{8, 16, 24, 32, 40, 48, 56, 64\}$, freeze the backbone, and train a linear classifier on ImageNet-1K at 256×256 resolution following Section A.5.1.

Table S.16: **Elastic computation for ImageNet-1K classification.** We report frozen-backbone linear-probe top-1 accuracy. C denotes the number of active core tokens at inference time.

C	VECA-S	VECA-S+	VECA-B	VECA-L
64	74.63	77.03	82.18	84.51
56	74.22	76.97	82.18	84.27
48	74.04	76.97	82.19	84.35
40	73.89	76.97	82.11	84.23
32	73.73	77.03	82.02	84.23
24	73.52	76.65	81.87	84.15
16	72.87	76.17	81.54	83.90
8	71.30	74.39	80.45	83.06

Results. Classification remains stable across a broad range of core-token budgets. For VECA-B, reducing the budget from $C = 64$ to $C = 32$ changes accuracy by only 0.16 points, while $C = 8$ retains 80.45% accuracy. The small non-monotonic differences at larger budgets are within normal linear-probe variation.

A.12.3 Elastic Computation: Dense Prediction

Protocol. We use the frozen-backbone protocol in Section A.5 and vary the same active core-token budgets. We evaluate semantic segmentation on PASCAL VOC and ADE20K and depth estimation on NYUv2 at 512×512.

Table S.17: **Elastic computation for dense prediction.** Each cell reports PASCAL VOC mIoU, ADE20K mIoU, and NYUv2 RMSE, respectively. C denotes the number of active core tokens at evaluation time.

C	VECA-S	VECA-S+	VECA-B	VECA-L
64	81.56 / 44.82 / 0.4347	84.12 / 46.55 / 0.4252	87.07 / 50.69 / 0.3705	88.62 / 54.17 / 0.3379
56	81.45 / 44.80 / 0.4352	83.92 / 46.24 / 0.4259	86.94 / 50.40 / 0.3751	88.62 / 54.17 / 0.3398
48	81.51 / 44.53 / 0.4361	83.94 / 46.08 / 0.4270	86.91 / 50.63 / 0.3748	88.59 / 54.09 / 0.3407
40	81.40 / 43.87 / 0.4382	83.84 / 45.71 / 0.4299	86.85 / 50.53 / 0.3754	88.51 / 53.81 / 0.3412
32	81.00 / 43.98 / 0.4430	83.84 / 45.56 / 0.4327	86.72 / 49.49 / 0.3789	88.37 / 53.90 / 0.3432
24	81.06 / 43.23 / 0.4476	83.43 / 44.60 / 0.4375	86.34 / 49.44 / 0.3829	87.78 / 53.65 / 0.3474
16	80.40 / 41.20 / 0.4562	82.60 / 43.03 / 0.4453	85.37 / 47.52 / 0.3965	87.67 / 52.20 / 0.3563
8	77.56 / 37.48 / 0.4975	79.18 / 38.96 / 0.4898	83.84 / 44.51 / 0.4330	86.50 / 48.10 / 0.3956

Results. Table S.17 shows that dense prediction is more sensitive to the active token budget. Performance degrades gradually as C decreases, with larger drops at small budgets, particularly on ADE20K and NYUv2. For example, reducing VECA-B from $C = 64$ to $C = 32$ changes ADE20K mIoU from 50.69 to 49.49 and NYUv2 RMSE from 0.3705 to 0.3789.

Together, the two settings highlight VECA's elastic computation behavior. A small token budget preserves much of the global information needed for classification, while additional core tokens provide finer spatial information that is more important for dense prediction.

A.13 CROSS-FRAME SEMANTIC TRANSFERABILITY

This section provides implementation details to test whether the semantic structure captured by VECA cores can be reused across video frames. Given the annotated first frame of a DAVIS-2017 sequence, we first associate its representation with object labels. We then use this labeled representation to predict the object labels of patches in later frames.

Assigning object labels to the first-frame representation. Let the final-layer patch features of the annotated first frame be

$$P_1 \in \mathbb{R}^{N\times D},$$

and let its VECA cores be

$$R_1 \in \mathbb{R}^{C\times D}.$$

The ground-truth mask is converted to patch-level soft labels

$$Y_1 \in [0,1]^{N\times L},$$

where L contains the background and object identities in the sequence. For example, with two objects, each patch has a three-dimensional soft label over background and the two objects.

To assign labels to the cores, we treat the cores as queries and the first-frame patches as keys:

$$Q = R_1, \qquad K = P_1, \qquad V = Y_1.$$

After ℓ_2-normalizing the feature vectors, we compute cosine-similarity weights

$$A(Q,K) = \mathrm{softmax}_k\left(\frac{\bar{Q}\bar{K}^\top}{\tau}\right),$$

where $\mathrm{softmax}_k$ keeps, for each row, only the k largest entries and sets the rest to $-\infty$ before normalization. For core labeling, we use $k = 64$, so each core attends to its 64 most similar first-frame patches; this neighbor count is independent of the core budget C. The corresponding soft label of each core is

$$Y_{\mathrm{core}} = A(R_1, P_1)Y_1 \in [0,1]^{C\times L}.$$

Thus, each core receives a soft object label by aggregating the labels of the patches most similar to it. This is a post-hoc feature-similarity readout and does not use VECA's internal attention weights.

Propagating labels to later frames. For a later frame t, we run VECA normally and take its final-layer patch features

$$P_t \in \mathbb{R}^{N_t\times D}.$$

The current patches are used as queries, while the stored first-frame cores and their assigned labels form the reference:

$$Q = P_t, \qquad K = R_1, \qquad V = Y_{\mathrm{core}}.$$

We again compute

$$A(P_t, R_1) = \mathrm{softmax}\left(\frac{\bar{P}_t\bar{R}_1^\top}{\tau}\right),$$

over all cores, and predict the labels of the current-frame patches as

$$\hat{Y}_t = A(P_t, R_1)Y_{\mathrm{core}}.$$

Each current patch therefore retrieves object labels from the first-frame cores according to feature similarity. The predicted patch labels are upsampled to the original image resolution to obtain the segmentation mask. The cores produced by frame t are not used in this retrieval, and no correspondence between core indices across frames is assumed.

Reference representations. We use the same label-transfer rule for all representations and change only the first-frame representation used as keys and values. All compact references are labeled with the same top-64 readout used for VECA cores.

Dense patches retain all first-frame patches:

$$K = P_1, \qquad V = Y_1.$$

Uniform sampling retains 64 spatially distributed first-frame patches as keys.

K-means centroids cluster the ℓ_2-normalized first-frame patch features P_1 into 64 clusters with K-means++ initialization, and use the centroids as keys.

VECA-CoreOnly uses the same procedure as VECA cores, but with $C = 256$ cores from the CoreOnly ablation, in which dense patches bypass the intermediate blocks and only the cores are updated in those layers.

VECA cores use $C = 64$ final cores from the standard VECA model, with their labels assigned from the first-frame patches as described above.

Implementation details. We use $\tau = 0.07$ throughout. For the dense reference, each current patch retrieves from its 20 most similar first-frame patches. For reduced-token representations, all retained tokens are used when propagating labels to later frames. We report the standard DAVIS-2017 $\mathcal{J}$, $\mathcal{F}$, and $\mathcal{J}\&\mathcal{F}$ metrics on the validation split.