PRISM: PROGRESSIVE REASONING THROUGH ITERATIVE SLOT MEMORY FOR VISION

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern vision models process images in a single feed-forward pass, which limits their ability to recover missing evidence or refine uncertain representations under incomplete observations. Inspired by the iterative nature of human perception, we introduce PRISM (Progressive Reasoning through Iterative Slot Memory), a pyramid vision architecture that reasons over images through iterative refinement. At a high level, PRISM groups visual features into object-centric representations, retrieves relevant patterns from a learned memory, and iteratively refines the representation to resolve ambiguity and recover missing information. This organize–recall–refine process operates recurrently across multiple scales, enabling progressive improvement of visual representations. Across standard vision tasks, including image classification, object detection, and semantic segmentation, PRISM achieves competitive performance while demonstrating improved robustness under incomplete observations such as occlusion. These results suggest that iterative reasoning with structured representations and memory is a promising direction for building more resilient and adaptive vision models. Source code and models will be released.

1 INTRODUCTION

Visual recognition systems Vaswani et al. (2017); Liu et al. (2021b); Wang et al. (2021; 2022); Shi et al.; Jia et al. (2025); Talbot et al. (2023); Cai et al. (2025) are expected to operate reliably in real-world environments, where observations are often incomplete due to occlusion or missing visual evidence. Although modern deep networks achieve impressive performance on clean benchmarks, their representations can become fragile when important visual cues are unavailable Lin et al. (2025); Wang et al. (2024); Zhang et al. (2020); Bomatter et al. (2021); Liu et al. (2022). A key limitation is their predominantly feed-forward nature: visual features are computed in a single pass and propagated through layers without revisiting or refining intermediate representations. As a result, incomplete information is compounded in deeper layers, leading to degraded recognition rather than progressive inference.

A long-standing goal in representation learning is to model scenes in terms of object-centric entities Locatello et al. (2020); Wu et al. (2023); Wang et al. (2023); Han et al. (2024), such as objects and parts, rather than undifferentiated pixel collections or dense feature tokens arranged on rigid grids in transformer-based feature maps. Object-centric representations improve robustness by aggregating information across regions belonging to the same entity, allowing partial observations to remain interpretable through structured object-level evidence and enabling inference of occluded regions from visible parts.

However, object-centric grouping alone is not sufficient. Methods such as slot attention Wang et al. (2024; 2023); Han et al. (2024) organize features into coherent entities, but remain fundamentally bottom-up: the resulting slots reflect only the available evidence and become incomplete or ambiguous under occlusion. Without access to prior knowledge, a partial slot with degraded visual information due to occlusion cannot reliably infer the full structure of an object or disambiguate between competing interpretations.

Conversely, relying on a prototype memory alone (e.g., VQ-VAEvan den Oord et al. (2017)) is also insufficient. When applied directly to dense feature tokens, the inputs are entangled mixtures of multiple objects and background, leading to ambiguous or noisy prototype matching. As a result, the

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.

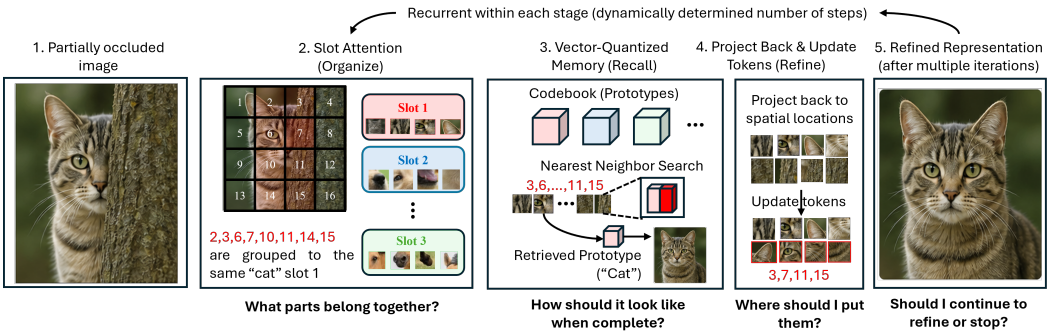


Figure 1: **Schematic of Progressive Reasoning through Iterative Slot Memory (PRISM).** During inference, the right half of the cat is occluded by a tree. PRISM performs progressive reasoning as follows: (1) visual tokens with similar semantics and structure are grouped into an object-centric "cat" slot; (2) the partial slot, degraded by occlusion, is matched to the nearest prototype in a learned memory during training, retrieving a complete cat pattern; (3) the retrieved prototype is projected back to the feature space to update tokens associated with the cat; and (4) this process is iterated for a dynamically determined number of steps until the representation becomes complete and coherent. This organize–recall–refine cycle is applied recurrently to progressively improve the representation.

learned codebook tends to capture blurred or mixed patterns rather than clean object-level structures, making retrieval unreliable for completion.

This motivates combining both components. A vector-quantized memory provides global priors in the form of prototypical patterns, enabling restoration of missing features via top-down modulation, while slot attention supplies the necessary structure by grouping tokens into object-centric representations that are suitable for prototype matching. Together, they play complementary roles: slot attention organizes visual evidence, and the memory module enables completion by grounding these representations in learned prototypes. This synergy allows the model to both organize and recover visual information, overcoming the limitations of feed-forward architectures that lack mechanisms to organize, recall, and refine representations under incomplete observations.

In parallel, biological vision exhibits recurrent dynamics across the cortical hierarchy (e.g., V1–V2–V4), with extensive behavioral and neurophysiological evidence showing that perception is iteratively refined through feedback and lateral interactions Zhang et al. (2020); Tang et al. (2018); Kietzmann et al. (2019); Kubilius et al. (2019); Kar & DiCarlo (2021) Loosely inspired by neuroscientific and behavioral evidence from biological vision systems, we propose PRISM, a pyramid vision architecture that introduces progressive reasoning through iterative slot memory.

As illustrated in **Figure. 1**, PRISM first groups visual tokens into object-centric representations that capture the underlying visual structure. These slots are then mapped into a vector-quantized memory space, where a learned codebook stores prototypical slot patterns acquired during training. When observations are incomplete, the corresponding slot representations may also be degraded. PRISM retrieves nearby prototypes from memory to restore these slots and projects the restored information back to their spatial locations to update token features.

This organize–recall–refine process is applied recurrently within each stage, enabling progressive refinement of representations over multiple iterations. Moreover, these recurrent updates are adaptive within each stage: PRISM dynamically determines the number of refinement iterations, allowing more challenging inputs to receive additional reasoning steps while keeping simpler cases efficient. Together, these properties enable PRISM to treat visual inference not as a single feed-forward transformation, but as a recurrent process of progressively organizing visual evidence, recalling learned prototypes, and refining internal representations.

We evaluate PRISM on standard computer vision tasks, including image classification, object detection, and semantic segmentation. Across these settings, PRISM achieves competitive performance while demonstrating improved robustness under partial observations and missing visual evidence. These results highlight iterative slot memory as a general architectural principle for strengthening visual recognition beyond task-specific designs. The contributions of this paper are as follows:

1. We introduce a recoverable representation paradigm, where visual features are organized into object-centric abstractions and grounded in a learned prototype space, enabling reconstruction of coherent representations from incomplete observations.

2. We propose a recurrent, stage-wise adaptive refinement mechanism, where visual representations are iteratively updated through organize–recall–refine cycles, with the number of iterations dynamically determined and all tokens jointly updated at each step.

3. We show that prototype-based restoration combined with dynamic recurrent refinement provides a strong inductive bias for artificial vision systems, where representations are progressively refined rather than produced in a single feed-forward pass, yielding high efficiency and improved robustness under occlusion.

2 Related Works

Slot-Based and Object-Centric Representations. Prior work models scenes via object-centric representations by decomposing inputs into latent entities. Slot Attention and its variants group features into slots through iterative competition Locatello et al. (2020); Wang et al. (2023; 2024); Han et al. (2024), and have been widely used in unsupervised object discovery and scene decomposition (e.g., MONet Burgess et al. (2019), IODINE Greff et al. (2019)). Extensions to video further introduce temporal consistency in object-centric representations (e.g., SAVi Kipf et al. (2022), SlotFormer Wu et al. (2023), VideoSAUR Zadaianchuk et al. (2023)). However, these methods are typically standalone or auxiliary modules and are rarely integrated into high-performance multi-stage backbones for visual recognition trained on large-scale datasets, such as ImageNet Deng et al. (2009). Moreover, continuous slots remain sensitive to incomplete observations, leading to inconsistent representations. In contrast, PRISM embeds slot-based grouping within a pyramidal architecture and augments it with a discrete prototype memory, enabling stable and recoverable representations. It performs on par with competitive baselines on large-scale ImageNet recognition benchmarks and achieves superior performance on occluded ImageNet images without any prior exposure to occluded samples during training.

Discrete Representation Learning. Vector quantization maps continuous features to a finite set of learned prototypes. VQ-VAE van den Oord et al. (2017) and VQGAN Esser et al. (2021) demonstrate their effectiveness for image generative modeling, with subsequent works introducing improvements such as residual quantization Lee et al. (2022), hierarchical quantization Takida et al. (2024), and lookup-free quantization Yu et al. (2024). These methods show that discrete codebooks provide compact and reusable visual representations. However, prior work has primarily used vector quantization for generative modeling or tokenization, rather than for restoration within discriminative visual recognition models. Moreover, applying VQ to dense token grids can entangle multiple objects, resulting in ambiguous prototype assignments. In contrast, PRISM performs quantization at the slot level, enabling prototype-based restoration of object-centric representations.

Recurrent and Adaptive Computation. Recurrent and adaptive computation methods enable models to dynamically adjust their computational depth, including ACT Graves (2016), Universal Transformers Dehghani et al. (2019), and PonderNet Banino et al. (2021). In vision, approaches such as A-ViT Yin et al. (2022) and depth-adaptive transformers Elbayad et al. (2020); Liu et al. (2021a) adjust inference depth based on input complexity. These methods primarily treat recurrence as variable-depth computation over dense feature representations. In contrast, PRISM grounds recurrence in prototype-based representation restoration, where each iteration explicitly refines features via vector-quantized memory, enabling structured and progressive recovery under incomplete observations. The superior performance of PRISM empirically supports this design.

3 PRISM

PRISM is a four-stage vision pyramid architecture (**Figure. 2**), where each stage begins with a 2D convolutional layer that converts the input image (at the first stage) or the incoming feature map (at subsequent stages) into a sequence of visual tokens. Across stages, the spatial resolution is progressively reduced by a factor of two, forming a standard feature pyramid with resolutions of $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$, and $\frac{1}{32}$ of the input image, while the channel dimension increases accordingly, enabling PRISM

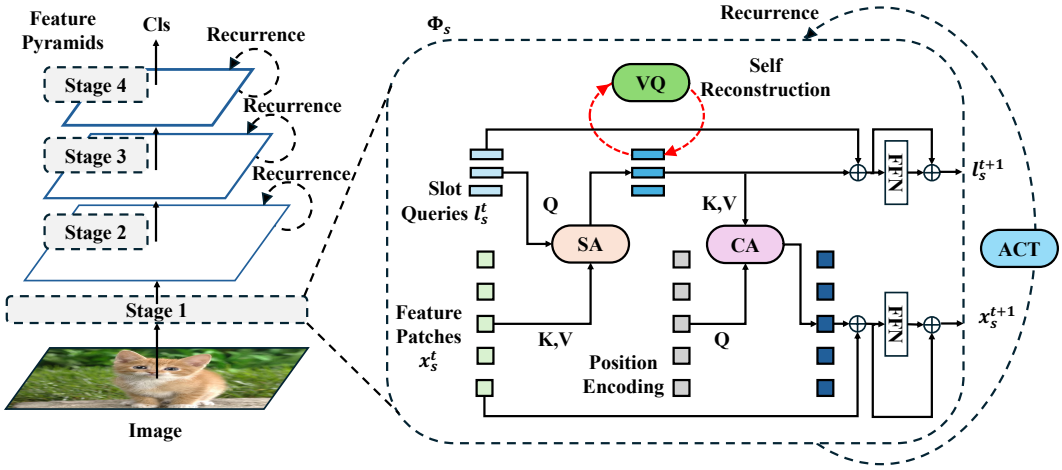


Figure 2: **PRISM Architecture.** PRISM consists of four stages in a feature pyramid, where each stage recurrently refines token features at different resolutions before passing them to the next stage, and the final representation is used for classification (Cls). Within each stage, feature tokens are first grouped into object-centric slots via Slot Attention (SA), which performs competitive grouping into a fixed number of slots to produce structured representations capturing distinct entities. These slot representations are grounded in a vector-quantized (VQ) memory trained with a self-reconstruction objective, where continuous features are mapped to a finite set of learned prototypes, yielding compact and reusable representations. The resulting prototypical slot features are then redistributed to their corresponding spatial locations through cross-attention (CA), where positional embeddings serve as queries (Q), enabling spatial information flow back to the feature map. The resulting feature maps are further processed by feed-forward networks (FFNs) with residual connections. We encapsulate these core operations within the s-th stage as a recurrent function denoted by Φ_s (the big dashed box). Each stage is executed recurrently via Φ_s to enable iterative refinement, with adaptive halting determining the number of recurrent steps through an Adaptive Computation Time (ACT) module.

to learn visual features at different levels of granularity. Unlike conventional pyramid backbones such as PVT (Wang et al., 2021), PVTv2 (Wang et al., 2022), and Swin Transformer (Liu et al., 2021b), which stack multiple feed-forward blocks within each stage, PRISM employs a single block per stage and applies it recurrently. See additional architecture details in **Appendix Section. A3**.

As shown in **Figure. 2**, at the recurrent step t of each stage s, PRISM takes feature tokens $\mathbf{x}_s^t \in \mathbb{R}^{N_s \times D_s}$, slot queries $\mathbf{l}_s^t \in \mathbb{R}^{K_s \times D_s}$, and positional embeddings $\mathbf{p}_s \in \mathbb{R}^{N_s \times D_s}$ as input, and builds on three core operations: slot attention (SA), vector quantization (VQ), and cross-attention (CA) to refine token representations. We denote these operations as $\mathrm{SA}(\cdot;\cdot;\cdot)$, $\mathrm{VQ}(\cdot;\cdot;\cdot)$, and $\mathrm{CA}(\cdot;\cdot;\cdot)$, respectively, and use them as fundamental building blocks throughout each stage of PRISM. Detailed mathematical formulations are provided in **Appendix Section. A1**. We use Φ_s to denote the recurrent function that encapsulates all these core operations within the s-th stage. Each stage is executed recurrently via Φ_s to enable iterative token refinement, with adaptive halting determining the number of recurrent steps through an Adaptive Computation Time (ACT) module based on input complexity. See the mathematical formulation of $\mathrm{ACT}(\cdot;\cdot)$ in **Appendix Section. A1**. Next, we introduce the three core modules, SA, VQ, and CA, within Φ_s, as well as the ACT mechanism in detail.

3.1 Grouping Semantically and Structurally Similar Tokens into Slot Entities

PRISM performs competitive grouping over patch features into a fixed number of slots to obtain structured representations capturing distinct entities. Slot features $\mathbf{l}_s^t$ serve as queries, while position-aware patch features $(\mathbf{x}_s^t + \mathbf{p}_s)$ and the original patch features $\mathbf{x}_s^t$ serve as keys and values:

$$\mathbf{v}_{s,\mathrm{grp}}^t = \mathrm{SA}\big(W_{q,s}\mathbf{l}_s^t, W_{k,s}(\mathbf{x}_s^t + \mathbf{p}_s), W_{v,s}\mathbf{x}_s^t\big)\,. \tag{1}$$

$W_{q,s}, W_{k,s}$ and $W_{v,s}$ are learnable linear projection matrices. Unlike classical Slot Attention Locatello et al. (2020), we incorporate positional modulation in the keys, encouraging slot features to capture not only semantic content but also spatial structure of object entities. Therefore, in addition to the grouped values $\mathbf{v}_{s,\mathrm{grp}}^t$, we also compute grouped keys $\mathbf{k}_{s,\mathrm{grp}}^t$, which are used in the subsequent redistribution step to place object-centric representations back at correct spatial locations.

Let $\mathbf{W}^t_{s,\mathrm{grp}}$ denote the attention weights measuring similarity between slot queries and position-aware keys produced by SA (see also the formulation of W_{ij} in **Appendix Equation. 11**). The grouped keys are obtained by weighting the position-aware keys with these attention weights:

$$\mathbf{k}^t_{s,\mathrm{grp}} = \mathbf{W}^t_{s,\mathrm{grp}} W_{k,s}(\mathbf{x}^t_s + \mathbf{p}_s). \tag{2}$$

3.2 Vector-Quantized Memory for Storing Learned Prototypes

During training, PRISM learns a fixed set of compact prototypes representing object-centric features from clean images with complete visual information, and stores them in a vector-quantized memory. During inference, these prototypical slot features are used to correct partial slot representations arising from incomplete observations. We adopt the standard VQ-VAE formulation van den Oord et al. (2017), and define a stage-wise codebook as $\mathbf{E}_s$.

Instead of quantizing $\mathbf{k}^t_{s,\mathrm{grp}}$ and $\mathbf{v}^t_{s,\mathrm{grp}}$ separately, we first concatenate them and map the result into a compact latent space with the encoder $\mathbf{Enc}_s$, which is then quantized using VQ and decoded back to the grouped representations with the decoder $\mathbf{Dec}_s$:

$$[\hat{\mathbf{k}}^t_{s,\mathrm{grp}}, \hat{\mathbf{v}}^t_{s,\mathrm{grp}}] = Dec_s\big(\mathrm{VQ}\big(Enc_s\big([\mathbf{k}^t_{s,\mathrm{grp}}, \mathbf{v}^t_{s,\mathrm{grp}}]\big)\,;\mathbf{E}_s\big)\big)\,. \tag{3}$$

This module serves as a prototype memory over grouped slot representations. During training, it is supervised by two objectives: (1) a reconstruction loss that maps slot representations into the codebook space and reconstructs them via the decoder, and (2) a commitment loss that encourages encoder outputs to remain close to their assigned codebook vectors, stabilizing training and preventing drift in the continuous feature space:

$$\mathcal{L}_{s,\mathrm{vq}} = \sum_{t=1}^{T} \left[\left\| [\mathbf{k}^t_{s,\mathrm{grp}}, \mathbf{v}^t_{s,\mathrm{grp}}] - [\hat{\mathbf{k}}^t_{s,\mathrm{grp}}, \hat{\mathbf{v}}^t_{s,\mathrm{grp}}] \right\|_2^2 + \beta \left\| \mathbf{z}^t_s - \mathrm{sg}[\mathbf{z}^t_{s,\mathrm{vq}}] \right\|_2^2 \right], \tag{4}$$

where $\mathrm{sg}[\cdot]$ denotes the stop-gradient operator, $\mathbf{z}^t_s$ is the encoded feature vector, and $\mathbf{z}^t_{s,\mathrm{vq}}$ is its quantized representation in the codebook (see **Appendix Section. A1.2** for formulation).

During inference, grouped representations are passed through the same memory to retrieve nearest prototypes, improving robustness and coherence under incomplete or corrupted observations. The reconstructed grouped key $\hat{\mathbf{k}}^t_{s,\mathrm{grp}}$ and value $\hat{\mathbf{v}}^t_{s,\mathrm{grp}}$ are then used in the subsequent redistribution step. In **Appendix Section. A5**, we provide detailed analysis of this VQ prototype memory.

3.3 Redistributing Restored Slot Features back to Their Original Spatial Locations

Next, $\hat{\mathbf{k}}^t_{s,\mathrm{grp}}$ and $\hat{\mathbf{v}}^t_{s,\mathrm{grp}}$ are redistributed to their original spatial locations. We use a cross-attention mechanism CA, where position embeddings for each spatial location on the feature map serve as queries, while the restored grouped slot keys and values serve as the key–value pairs:

$$\mathbf{x}^t_{s,\mathrm{back}} = \mathrm{CA}\Big(W_{q,s}\mathbf{p}_s, \hat{\mathbf{k}}^t_{s,\mathrm{grp}}, \hat{\mathbf{v}}^t_{s,\mathrm{grp}}\Big)\,. \tag{5}$$

In this way, $\hat{\mathbf{k}}^t_{s,\mathrm{grp}}$ encodes spatially aware information containing "what-to-where", aligning with positional embeddings to guide the redistribution of $\hat{\mathbf{v}}^t_{s,\mathrm{grp}}$ back to the correct spatial locations. Finally, $\hat{\mathbf{k}}^t_{s,\mathrm{grp}}$ and $\mathbf{x}^t_{s,\mathrm{back}}$ are passed through two independent FFNs with residual connections and MLP updates to produce the slot queries $\mathbf{l}^{t+1}_s$ and input tokens $\mathbf{x}^{t+1}_s$ for the next recurrent step $t+1$.

3.4 Within-Stage Dynamic Halting for Recurrence Control

The Adaptive Computation Time (ACT) mechanism dynamically controls when to stop recurrent refinement within each stage by predicting a global halting probability over the patch features. If the accumulated probability exceeds 1, the recurrence stops; otherwise, the process continues until this condition is satisfied. ACT allows simpler inputs to terminate early while allocating more computation to harder or more ambiguous cases. This leads to improved computational efficiency

Table 1: **ImageNet-1K classification under clean and occlusion settings.** We report parameters, FLOPs, and Top-1 accuracy on clean images and five occlusion settings. All experiments have three runs. Standard deviations in Top-1 accuracy for PRISM are reported in brackets (last row). See **Section. 4.1** for experimental details. Best results are bolded.

Model	Params	FLOPs	Clean	Patch Mask 0.6	Patch Mask 0.8	Block Mask 112	Only One 112	Only One 56
PVTv2-B1Wang et al. (2022)	13M	2.1G	78.7	53.8	39.6	74.6	62.8	27.8
XCiT-T24Ali et al. (2021)	12M	2.3G	79.4	53.1	41.2	76.5	61.5	25.6
BiXT-Ti/16Hiller et al. (2024)	15M	1.7G	80.1	55.6	40.5	77.2	64.2	28.5
BiFormer-TZhu et al. (2023)	13M	2.2G	**81.4**	60.4	42.1	77.2	66.4	30.4
PRISM	11M	2.2G	80.3 ±0.12	**69.1** ±0.18	**44.6** ±0.21	**78.9** ±0.10	**69.2** ±0.16	**33.8** ±0.25

and also acts as an implicit regularizer by preventing unnecessary iterative refinement, which helps reduce overfitting.

Formally, each stage is governed by the recurrent refinement function Φ_s, which jointly processes patch features $\mathbf{x}_s^t$ and slot queries $\mathbf{l}_s^t$, producing updated representations $\mathbf{x}_s^{t+1}$ and $\mathbf{l}_s^{t+1}$ at each step t. Since the halting decision depends only on the patch features, we omit slot queries in the notation for simplicity and write the stage output as:

$$\hat{\mathbf{x}}_s = \mathrm{ACT}(\mathbf{x}_s; \Phi_s), \tag{6}$$

We use the standard ponder loss for training the ACT module. This loss encourages PRISM to assign higher probability to earlier halting steps, thereby favoring shorter computation when possible. We set the maximum number of recurrent steps to $T_{\max} = 5$ for each stage.

$$\mathcal{L}_{\mathrm{ponder},s} = \sum_{t=1}^{T_{\max}} t\, \hat{p}_s^t. \tag{7}$$

3.5 Training and Implementation Details

The overall training objective for PRISM is:

$$\mathcal{L} = \mathcal{L}_{\mathrm{cls}} + \lambda_{\mathrm{ponder}} \sum_{s=1}^{4} \mathcal{L}_{\mathrm{ponder},s} + \lambda_{\mathrm{vq}} \sum_{s=1}^{4} \mathcal{L}_{\mathrm{vq},s}, \tag{8}$$

where $\lambda_{\mathrm{ponder}} = 0.005$ and $\lambda_{\mathrm{vq}} = 0.01$ are the corresponding loss weights, and $\mathcal{L}_{\mathrm{cls}}$ denotes the cross-entropy classification loss.

We train PRISM from scratch for 300 epochs on 4 NVIDIA RTX A6000 GPUs with a total batch size of 1024. Following common ImageNet training practices Wang et al. (2022); Touvron et al. (2021), we apply standard data augmentation techniques during training. Optimization is performed using AdamW with an initial learning rate of 1×10^{-3}, $\beta_1 = 0.9$, and $\beta_2 = 0.999$, together with a cosine learning-rate schedule. Images are resized and cropped to 224×224 during training, and center cropping is used during validation.

4 Experiments

We evaluate PRISM on image classification, object detection, and semantic segmentation under clean and occluded settings. To benchmark performance, we compare PRISM against strong vision backbones, including PVTv2, BiXT, BiFormer, and DeBiFormer across all tasks. All models are trained without occluded images and evaluated on them only at test time to assess out-of-distribution robustness. For detection and segmentation tasks, we initialize the backbone with PRISM pretrained on ImageNet-1K Deng et al. (2009) and follow standard fine-tuning protocols Wang et al. (2022).

4.1 Image Classification

Dataset. We evaluate image classification on ImageNet-1K Deng et al. (2009) using the standard training and validation splits. **Image Occlusion.** To assess robustness under incomplete observations,

we introduce five occlusion settings on ImageNet-1K images of size 224×224: random patch masking with ratios of 0.6 and 0.8 (PatchMask0.6 and PatchMask0.8), block occlusion with a region of size 112×112 (BlockMask112), and single-visible-patch evaluation retaining only one 112×112 or 56×56 region (OnlyOne112 and OnlyOne56). These settings simulate varying levels and spatial patterns of missing information. **Metrics.** We report model complexity and performance using number of parameters, FLOPs, and Top-1 accuracy.

Results. We report image classification results in **Table. 1**. All models have comparable computational complexity with similar FLOPs. PRISM achieves competitive performance on clean images and substantially stronger robustness under all occlusion settings.

On clean images, PRISM achieves a Top-1 accuracy of 80.3%, outperforming PVTv2-B1 Wang et al. (2022) by 1.6% while using fewer parameters (11M vs. 13M). All models experience performance degradation under occlusion, highlighting the difficulty of recognition under incomplete visual evidence. Nevertheless, PRISM consistently surpasses all competitive baselines, with the advantage becoming more pronounced as occlusion severity increases. For example, under PatchMask$_{0.6}$, PRISM achieves 69.1%, outperforming BiFormer-T by 8.7% and PVTv2-B1 by 15.3%. Under the more challenging PatchMask$_{0.8}$ setting, PRISM reaches 44.6%, compared to 42.1% for BiFormer-T and 39.6% for PVTv2-B1. Similar trends are observed under structured occlusion. For instance, under OnlyOne$_{112}$, PRISM achieves 69.2%, exceeding BiFormer-T by 2.8% and PVTv2-B1 by 6.4%. These results suggest that PRISM not only improves clean-image recognition, but also maintains more reliable representations when large portions of the image are missing, demonstrating the effectiveness of iterative slot memory for robust recognition under occlusions. We also present extended robustness evaluation in **Appendix Section. A6** and additional comparisons in **Appendix Section. A8**.

4.2 Object Detection and Instance Segmentation

Dataset. We evaluate object detection and instance segmentation on the MSCOCO dataset. **Implementation Details.** Following standard transfer learning protocols, PRISM is initialized with weights pretrained on ImageNet-1K and fine-tuned on COCO for detection and instance segmentation. We integrate PRISM into Mask R-CNN and RetinaNet frameworks with a $1\times$ schedule. All models are fine-tuned on 4 NVIDIA RTX A6000 GPUs with a total batch size of 8. We train and evaluate PRISM and all baselines under the same detection framework for fair comparison. **Image Occlusion.** To evaluate robustness under incomplete object observations, we construct occluded COCO validation images by randomly selecting two object instances per image and applying a block mask to each instance. The mask removes either 30% or 50% of the instance area, resulting in two settings (M30 and M50) with different levels of missing visual evidence. This protocol tests whether models can recognize and localize objects when instance-level information is partially occluded. **Metrics.** For RetinaNet, we report bounding-box mAP. For Mask R-CNN, we report mAP^b for bounding-box detection and mAP^m for instance segmentation.

Results. We report object detection and instance segmentation results under both clean and occluded settings in **Table. 2(a)**. PRISM achieves competitive clean-image performance while substantially outperforming all baselines under occlusion, despite using significantly fewer parameters.

For RetinaNet, PRISM achieves 43.6 mAP on clean images, outperforming PVTv2-B1 Wang et al. (2022) by 2.4 points and PaCa-Ti Grainger et al. (2023) by 1.6 points, while remaining competitive with DaViT-T Ding et al. (2022) using roughly half the number of parameters. Under occlusion, the advantage of PRISM becomes more pronounced. PRISM achieves 38.8 mAP under M30 and 34.2 mAP under M50, surpassing DaViT-T by 1.9 and 2.7 points, respectively, indicating stronger robustness when object regions are partially missing.

For Mask R-CNN, PRISM also achieves strong detection and instance segmentation performance despite having the fewest parameters among all models. On clean images, PRISM achieves 44.5 mAP^b and 41.1 mAP^m, matching DaViT-T in mask accuracy while using approximately 40% fewer parameters. Under occlusion, the gains become larger. For example, under M30, PRISM reaches 40.6 mAP^b and 36.9 mAP^m, improving over DaViT-T by 2.0 and 1.6 points, respectively. Under the more challenging M50 setting, PRISM further achieves 37.2 mAP^b and 33.7 mAP^m, outperforming DaViT-T by 4.2 and 3.9 points. These results suggest that PRISM improves not

Table 2: **Downstream task performance under clean and occlusion settings.** (a) MSCOCO object detection and instance segmentation. Parameter count is reported in Column 2. In Row 2, C denotes the clean setting, and M30/M50 denote occlusion settings. Superscripts b and m indicate the metric type: for Mask R-CNN, bounding box detection (mAP^b) and mask segmentation (mAP^m) are reported, while no superscript denotes mAP with RetinaNet. See **Section. 4.2** for experimental details. (b) ADE20K semantic segmentation. Parameter count (Params.) and FLOPs are reported in Columns 2 and 3. C and Occ. denote the clean and occlusion settings. Model performance is reported in mIoU. See **Section. 4.3** for experimental details. Best are bolded. ('–') indicates that the model checkpoint is not officially provided and their result is therefore unavailable.

(a) MSCOCO detection and instance segmentation

Model	RetinaNet				Mask R-CNN						
	Param	C	M30	M50	Param	C^b	C^m	$M30^b$	$M30^m$	$M50^b$	$M50^m$
PVTv2-B1Wang et al. (2022)	23.8M	41.2	32.8	27.6	33.7M	41.8	38.8	34.1	31.1	28.7	25.9
PaCa-TiGrainger et al. (2023)	–	–	–	–	32.0M	43.3	39.6	37.5	34.7	31.8	28.7
DaViT-TDing et al. (2022)	38.5M	**44.0**	36.9	31.5	47.8M	**45.0**	**41.1**	38.6	35.3	33.0	29.8
PRISM	20.5M	43.6	**38.8**	**34.2**	31.4M	44.5	40.8	**40.6**	**36.9**	**37.2**	**33.7**

(b) ADE20K semantic segmentation

Model	Param.	FLOPs	mIoU	
			C	Occ.
XCiT-T12Ali et al. (2021)	8M	-	39.9	32.2
PVTv2-B1Wang et al. (2022)	17.8M	34.2G	42.5	34.8
BiXT-Ti/16Hiller et al. (2024)	19M	31.8G	39.2	30.5
PRISM	15.5M	35.6G	**44.8**	**39.7**

Table 3: **Ablation studies of PRISM on ImageNet-1K classification.** We report Top-1 accuracy on clean images and three occlusion conditions. Our default PRISM uses a maximum of 5 halting steps in (a), includes full VQ memory in (b), and employs dynamic halting (Dyn) in (c). Best are bolded.

	(a) Max halting steps during training			(b) Remove VQ during inference		(c) Fixed recurrent steps during inference					
Model Variants	4	5	6	w/o VQ	Full	1	2	3	4	5	Dyn
Clean	80.0	80.3	**80.4**	80.2	**80.3**	78.0	79.2	80.2	80.3	80.3	**80.3**
PatchMask0.8	43.7	44.6	**44.8**	37.8	**44.6**	31.4	38.9	43.1	44.0	44.2	**44.6**
BlockMask112	78.4	78.9	**79.0**	75.6	**78.9**	73.5	76.8	78.5	78.8	78.8	**78.9**
OnlyOne112	68.1	69.2	**69.4**	62.7	**69.2**	58.3	64.1	68.0	68.8	69.0	**69.2**

only classification robustness, but also object localization and instance-level segmentation under incomplete observations.

4.3 Semantic Segmentation

Dataset. We evaluate semantic segmentation on ADE20K. **Implementation Details.** Following standard transfer learning protocols, PRISM is initialized with weights pretrained on ImageNet-1K and fine-tuned on ADE20K. We select a standard semantic segmentation framework and fine-tune all models with the same framework for fair comparison. **Image Occlusion.** To evaluate robustness under incomplete observations, we construct occluded ADE20K validation images by randomly masking 30% of image patches. This setting removes scattered spatial evidence and tests whether models can maintain dense semantic predictions when parts of the scene are missing. **Metrics.** We report mean Intersection-over-Union (mIoU), which measures the average overlap between predicted and ground-truth semantic regions across all object classes.

Results. Results for PRISM and baseline models under clean and occluded settings are reported in **Table. 2(b)**. PRISM achieves the best semantic segmentation performance under both clean and occluded settings. On clean images, PRISM achieves an mIoU of 44.8, outperforming PVTv2-B1 Wang et al. (2022) by 2.3 points and BiXT-Ti/16 Hiller et al. (2024) by 5.6 points, despite using slightly fewer parameters. More importantly, PRISM maintains the best performance under occlusion with 39.7 mIoU, while PVTv2-B1 and BiXT-Ti/16 drop to 34.8 and 30.5, respectively. This corresponds to a gain of 4.9 points over PVTv2-B1 under random patch masking. The strong performance on dense prediction tasks suggests that PRISM preserves spatially coherent object-centric representations, enabling more reliable semantic grounding even when parts of the scene are missing.

4.4 Ablation Studies on PRISM

We conduct ablation studies of PRISM on ImageNet-1K classification under clean images and a subset of occlusion conditions described in **Section. 4.1**. Results in Top-1 accuracy are reported in **Table. 3**. **Effect of maximum halting steps.** Varying the maximum number of halting steps has limited impact on clean accuracy but significantly affects robustness under occlusion. Reducing the maximum steps from 5 to 4 slightly decreases clean accuracy ($80.3 \rightarrow 80.0$), but leads to larger drops

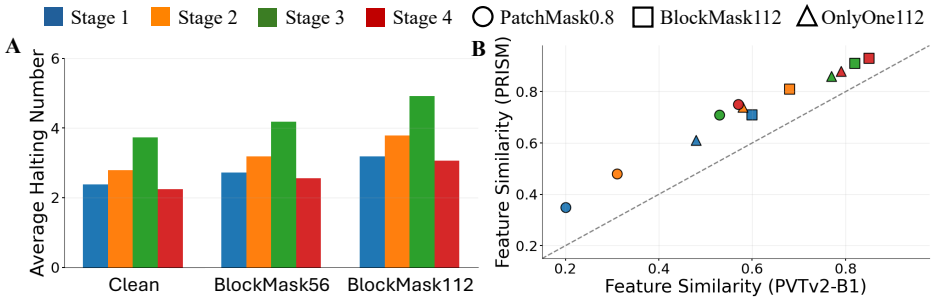


Figure 3: **Analysis of the halting dynamics of PRISM in image classification. A.** Average halting steps at each stage under clean and two occlusion settings. Colors denote different stages. **B.** Feature similarity between clean and occluded inputs, comparing PVTv2-B1 (x-axis) and PRISM (y-axis) across stages and occlusion settings. The dashed line denotes the diagonal. Each point corresponds to one stage (color) and one occlusion setting (marker). See **Section. 4.1** for details.

under occlusion (e.g., $69.2 \rightarrow 68.1$ on OnlyOne112). Increasing the limit from 5 to 6 yields only marginal gains (e.g., $44.6 \rightarrow 44.8$ on PatchMask0.8 and $69.2 \rightarrow 69.4$ on OnlyOne112), suggesting that 5 steps are sufficient for most inputs. **Effect of VQ memory.** Removing VQ memory at inference has negligible effect on clean accuracy ($80.3 \rightarrow 80.2$), but causes substantial degradation under occlusion, e.g., $44.6 \rightarrow 37.8$ on PatchMask0.8 and $69.2 \rightarrow 62.7$ on OnlyOne112. This highlights the importance of prototype-based restoration when visual evidence is incomplete. **Effect of dynamic halting.** Increasing the number of fixed recurrent steps improves performance (e.g., $31.4 \rightarrow 43.1$ on PatchMask0.8 and $58.3 \rightarrow 68.0$ on OnlyOne112 from $T{=}1$ to $T{=}3$), but saturates after 4–5 steps. Dynamic halting achieves 44.6 on PatchMask0.8 and 69.2 on OnlyOne112, outperforming fixed-step inference ($T{=}5$), demonstrating its ability to adapt computation to input difficulty.

4.5 Model Analysis

To probe the halting dynamics of PRISM, we analyze inference-time halting frequency across four stages of the visual hierarchy (**Figure. 3A**), as well as feature similarity between clean and occluded inputs (**Figure. 3B**). Also see an analysis of training-time halting frequencies in **Appendix Section. A2**. Inference-time halting frequency under occlusion in **Figure. 3A** shows that PRISM allocates more refinement steps as occlusion severity increases. For example, in Stage 3, the average halting steps increase from 3.73 (clean) to 4.18 (BlockMask56) and 4.92 (BlockMask112), indicating that more computation is required under harder conditions. Across the four stages of the visual hierarchy, Stage 3 consistently requires the most refinement, while Stage 4 remains lower, suggesting that intermediate stages are most responsible for feature restoration. Representation consistency results in **Figure. 3B** show that PRISM preserves higher feature similarity than PVTv2-B1 across all stages and occlusion settings. For instance, under PatchMask0.8, Stage 3 similarity increases from 0.53 (PVTv2-B1) to 0.71 (PRISM). This suggests that adaptive recurrence and the VQ memory help keep occluded features closer to their clean counterparts, improving model robustness and classification accuracy. We also provide a visualization example in **Appendix Section. A4** to explain how PRISM actually refines representations. See **Appendix Section. A7** for model efficiency.

5 Discussion

We present PRISM, a pyramid vision architecture that reasons over images through iterative slot memory. Instead of a single feed-forward pass, PRISM iteratively organizes patch features into slot-level entities, restores them via a vector-quantized prototype memory, and adapts the number of recurrent steps to input complexity via dynamic halting. Across image classification, object detection, and semantic segmentation, PRISM achieves competitive performance on standard benchmarks while showing stronger robustness under occlusion, suggesting that compositional abstraction, memory-based restoration, and adaptive recurrence are effective for robust visual recognition. Despite these benefits, the inference cost of PRISM may increase for challenging inputs requiring more refinement steps. Moreover, our evaluation primarily focuses on synthetic occlusions, and further validation under real-world occlusion scenarios remains important.

REFERENCES

Alaaeldin Ali, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. Xcit: Cross-covariance image transformers. *Advances in neural information processing systems*, 34:20014–20027, 2021.

Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder. In *Advances in Neural Information Processing Systems*, 2021.

Philipp Bomatter, Mengmi Zhang, Dimitar Karev, Spandan Madan, Claire Tseng, and Gabriel Kreiman. When pigs fly: Contextual reasoning in synthetic and natural scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 255–264, 2021.

Christopher P. Burgess, Loic Matthey, Nick Watters, Rishabh Kabra, Irina Higgins, Matthew Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. In *International Conference on Learning Representations*, 2019.

Yusen Cai, Bhargava Satya Nunna, Qing Lin, and Mengmi Zhang. Learning to see through a baby's eyes: Early visual diets enable robust visual intelligence in humans and machines. *arXiv preprint arXiv:2511.14440*, 2025.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019.

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.

Mingyu Ding, Bin Xiao, Noel Codella, Ping Luo, Jingdong Wang, and Lu Yuan. Davit: Dual attention vision transformers. In *European conference on computer vision*, pp. 74–92. Springer, 2022.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. In *International Conference on Learning Representations*, 2020.

Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12873–12883, 2021.

Ryan Grainger, Thomas Paniagua, Xi Song, Naresh Cuntoor, Mun Wai Lee, and Tianfu Wu. Paca-vit: learning patch-to-cluster attention in vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18568–18578, 2023.

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

Klaus Greff, Rapha"el Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2424–2433, 2019.

Shuangpeng Han, Ziyu Wang, and Mengmi Zhang. Flow snapshot neurons in action: Deep neural networks generalize to biological motion perception. *Advances in Neural Information Processing Systems*, 37:53732–53763, 2024.

Markus Hiller, Krista A Ehinger, and Tom Drummond. Perceiving longer sequences with bi-directional cross-attention transformers. *Advances in Neural Information Processing Systems*, 37:94097–94129, 2024.

Yanhao Jia, Ji Xie, S Jivaganesh, Hao Li, Xu Wu, and Mengmi Zhang. Seeing sound, hearing sight: Uncovering modality bias and conflict of ai models in sound localization. *arXiv preprint arXiv:2505.11217*, 2025.

Kohitij Kar and James J DiCarlo. Fast recurrent processing via ventrolateral prefrontal cortex is needed by the primate ventral stream for robust core visual object recognition. *Neuron*, 109(1): 164–176, 2021.

Tim C Kietzmann, Courtney J Spoerer, Lynn KA Sörensen, Radoslaw M Cichy, Olaf Hauk, and Nikolaus Kriegeskorte. Recurrence is required to capture the representational dynamics of the human visual system. *Proceedings of the National Academy of Sciences*, 116(43):21854–21863, 2019.

Thomas Kipf, Gamaleldin F. Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff. Conditional object-centric learning from video. In *International Conference on Learning Representations*, 2022.

Jonas Kubilius, Martin Schrimpf, Kohitij Kar, Rishi Rajalingham, Ha Hong, Najib Majaj, Elias Issa, Pouya Bashivan, Jonathan Prescott-Roy, Kailyn Schmidt, et al. Brain-like object recognition with high-performing shallow recurrent anns. *Advances in neural information processing systems*, 32, 2019.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11523–11532, 2022.

Qing Lin, Jingfeng Zhang, Yew-Soon Ong, and Mengmi Zhang. Make me happier: Evoking emotions through image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16367–16376, 2025.

Xiao Liu, Ankur Sikarwar, Gabriel Kreiman, Zenglin Shi, and Mengmi Zhang. Reason from context with self-supervised learning. *arXiv preprint arXiv:2211.12817*, 2022.

Yijin Liu, Fandong Meng, Jie Zhou, Yufeng Chen, and Jinan Xu. Faster depth-adaptive transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021a.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021b.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems*, volume 33, pp. 11525–11538, 2020.

Zenglin Shi, Jing Jie, Ying Sun, Joo Hwee Lim, and Mengmi Zhang. Unveiling the tapestry: the interplay of generalization and forgetting in continual learning, 2024. *URL https://arxiv.org/abs/2211.11174*.

Yuhta Takida, Wei-Hsiang Liao, Chihiro Takahashi, Takashi Shibuya, and Yuki Mitsufuji. Hq-vae: Hierarchical discrete representation learning with variational bayes. *Transactions on Machine Learning Research*, 2024.

Morgan B Talbot, Rushikesh Zawar, Rohil Badkundri, Mengmi Zhang, and Gabriel Kreiman. Tuned compositional feature replays for efficient stream learning. *IEEE Transactions on Neural Networks and Learning Systems*, 36(2):3300–3314, 2023.

Hanlin Tang, Martin Schrimpf, William Lotter, Charlotte Moerman, Ana Paredes, Josue Ortega Caro, Walter Hardesty, David Cox, and Gabriel Kreiman. Recurrent computations for visual pattern completion. *Proceedings of the National Academy of Sciences*, 115(35):8835–8840, 2018.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pp. 10347–10357. PMLR, 2021.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 568–578, 2021.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvt v2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, 8(3):415–424, 2022.

Ziyu Wang, Mike Zheng Shou, and Mengmi Zhang. Object-centric learning with cyclic walks between parts and whole. *Advances in Neural Information Processing Systems*, 36:9388–9408, 2023.

Ziyu Wang, Shuangpeng Han, and Mengmi Zhang. Pose prior learner: Unsupervised categorical prior learning for pose estimation. *arXiv preprint arXiv:2410.03858*, 2024.

Ziyi Wu, Nikita Dvornik, Klaus Greff, Thomas Kipf, and Animesh Garg. Slotformer: Unsupervised visual dynamics simulation with object-centric models. In *International Conference on Learning Representations*, 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*, 2022.

Lijun Yu, José Lezama, Nitesh Bharadwaj Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Agrim Gupta, Xiuye Gu, Alexander G. Hauptmann, Boqing Gong, Ming-Hsuan Yang, Irfan Essa, David A. Ross, and Lu Jiang. Language model beats diffusion – tokenizer is key to visual generation. In *International Conference on Learning Representations*, 2024.

Andrii Zadaianchuk, Matthaeus Kleindessner, Yi Zhu, Francesco Locatello, Thomas Brox, and Georg Martius. Object-centric learning for real-world videos by predicting temporal feature similarities. In *Advances in Neural Information Processing Systems*, volume 36, pp. 12710–12730, 2023.

Mengmi Zhang, Claire Tseng, and Gabriel Kreiman. Putting visual object recognition in context. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12985–12994, 2020.

Lei Zhu, Xinjiang Wang, Zhanghan Ke, Wayne Zhang, and Rynson WH Lau. Biformer: Vision transformer with bi-level routing attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10323–10333, 2023.

A Appendix

A1 Preliminaries

We introduce the basic operators used in our model, including attention-based feature aggregation, vector quantization, and adaptive computation.

A1.1 Cross-Attention (CA) and Slot-Attention (SA)

Given query, key, and value matrices $\mathbf{Q} \in \mathbb{R}^{N_q \times D}$, $\mathbf{K} \in \mathbb{R}^{N_k \times D}$, and $\mathbf{V} \in \mathbb{R}^{N_k \times D}$, the dot product $\mathbf{A}$ between $\mathbf{Q}$ and $\mathbf{K}$ is:

$$\mathbf{A} = \frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{D}} \in \mathbb{R}^{N_q \times N_k} \tag{9}$$

The output of a cross-attention operation, denoted by $\mathrm{CA}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$, is defined as:

$$\texttt{attn}_{i,j} = \frac{\exp^{A_{i,j}}}{\sum_{N_k} \exp^{A_{i,j}}}, \mathrm{CA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \texttt{attn} \times V \tag{10}$$

For Slot AttentionLocatello et al. (2020), we similarly write the grouping operation as $\mathrm{SA}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$. Its output is defined as:

$$\texttt{attn}_{i,j} = \frac{\exp^{A_{i,j}}}{\sum_{N_q} \exp^{A_{i,j}}}, \mathbf{W_{i,j}} = \frac{\texttt{attn}_{i,j}}{\sum_{N_k} \texttt{attn}_{i,j}}, \mathrm{SA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{W} \times V \tag{11}$$

Here, $\texttt{attn}_{i,j}$ normalizes over the slot dimension so that slots compete for each token, while $W_{i,j}$ further normalizes over the token dimension before aggregation.

A1.2 Vector Quantization (VQ)

Given an input feature matrix $\mathbf{Z} \in \mathbb{R}^{N \times D}$ and a codebook $\mathbf{E} = \{\mathbf{e}_m\}_{m=1}^{M}$ with $\mathbf{e}_m \in \mathbb{R}^D$, vector quantization maps each feature $\mathbf{z}_i$ to its nearest code:

$$m_i = \arg\min_m \|\mathbf{z}_i - \mathbf{e}_m\|_2^2, \qquad \mathbf{z}_{q,i} = \mathbf{e}_{m_i}. \tag{12}$$

The quantized output is denoted by $\mathbf{Z}_q = \mathrm{VQ}(\mathbf{Z}; \mathbf{E})$. We adopt the EMA-style codebook update in VQ-VAE van den Oord et al. (2017), where each code is updated by the moving average of the features assigned to it. In this way, the codebook tracks the centers of the assigned features during training.

A1.3 Adaptive Computation Time (ACT)

Given token features $\mathbf{X}$, we define the dynamic halting process as $\hat{\mathbf{X}} = \mathrm{ACT}(\mathbf{X}; \Phi)$, where $\mathbf{X}^t \in \mathbb{R}^{N \times D}$ denotes the token features at iteration t, and $\hat{\mathbf{X}} \in \mathbb{R}^{N \times D}$ denotes the final aggregated output. The features are updated recurrently by an updating function Φ:

$$\mathbf{X}^{t+1} = \Phi(\mathbf{X}^t). \tag{13}$$

Instead of predicting halting probabilities for individual tokens, an improved version of predicting single global halting probability for the whole token set at each iteration is introduced. Specifically, a global feature is first computed:

$$\bar{\mathbf{x}}^t = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n^t, \tag{14}$$

and then predict the halting variables by

$$p^t = \sigma(\mathrm{FC}(\bar{\mathbf{x}}^t)),\ T = \min\left\{ t : \sum_{t'=1}^{t} p^{t'} \geq 1 - \epsilon \right\},\ R = 1 - \sum_{t=1}^{T-1} p^t, \tag{15}$$

where $p^t \in (0, 1)$ is the global halting probability at iteration t, T is the halting step, and R is the remainder at the halting step. We set the hyperparmaeter $\epsilon = 0.01$ to improve numerical stability and enable smooth halting behavior by introducing a slightly relaxed stopping threshold, which helps stabilize the learning of the halting mechanism. The final output is computed as a weighted aggregation over recurrent states, where intermediate steps use their halting probabilities as weights, the final halting step uses the remainder, and steps after T receive zero weight. The aggregation over time steps is defined as

$$\hat{p}^t = \begin{cases} p^t, & t < T, \\ R, & t = T, \\ 0, & t > T, \end{cases} \qquad \hat{\mathbf{X}} = \sum_{t=1}^{T_{\max}} \hat{p}^t \mathbf{X}^t, \tag{16}$$

where $T_{\max}$ is the maximum number of iterations.

A2 Stage-wise Halting Dynamics during Training

As shown in **Figure. A1**, the halting numbers of all four stages increase rapidly in the early phase of training and become nearly stable after roughly the first one-third of optimization. This suggests that the adaptive computation policy is learned relatively early, after which the model mainly focuses on improving the quality of its representations rather than substantially changing how much computation

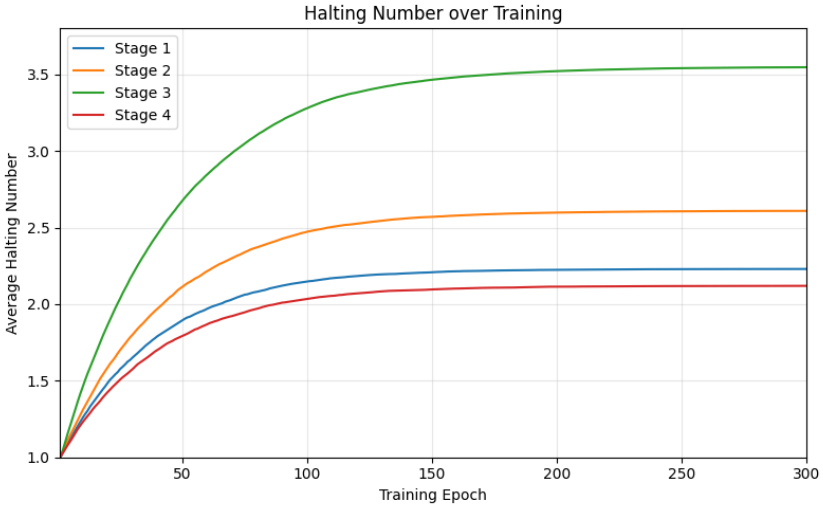


Figure A1: **Halting dynamics of PRISM.** Average halting numbers as a function of training epochs, showing how the model adaptively allocates recurrent refinement steps during training.

is allocated. In other words, the network first discovers *how much* iterative refinement each stage should perform, and then uses the remaining training process to improve *what* is computed within this recurrent budget.

A second important observation is that the learned halting numbers are strongly stage-dependent in the visual hierarchy of PRISM. Stage 3 converges to the largest halting number, followed by Stage 2, while Stage 1 and Stage 4 remain lower. This pattern is meaningful rather than incidental. The early stage mainly operates on relatively local patterns, where recurrent refinement is useful but still limited by the low-level nature of the features. The deepest stage (Stage 4), in contrast, already works on highly abstract representations, so fewer additional refinement steps are needed once the semantics have largely stabilized. The middle stages (Stages 2 and 3) therefore become the main locus of recurrent computation, where local evidence and global structure must be integrated into coherent object-level representations. This stage-wise allocation indicates that the halting mechanism does not collapse to a trivial constant-depth solution, but instead learns a structured computation pattern aligned with the semantic roles of the pyramid stages.

A3 Additional Architecture and Hyperparameter Details

PRISM consists of four hierarchical stages with embedding dimensions $[64, 128, 320, 512]$ and attention heads $[1, 2, 5, 8]$, respectively. Each stage uses eight slots and an independent codebook containing 64 prototype entries. The first stage uses a 4×4 patch embedding with stride 4, while subsequent stages downsample the spatial resolution by a factor of 2, resulting in cumulative downsampling rates of $[4, 8, 16, 32]$.

We further study the influence of the number of slots and codebook entries on ImageNet-1K. In each experiment, only the examined hyperparameter is varied while all other configurations remain at their default values. As shown in **Table A1**, using 4, 8, and 16 slots per stage gives 79.8%, 80.3%, and 80.2% Top-1 accuracy, respectively, with the codebook size fixed at 64. Using 32, 64, and 128 codebook entries gives 80.0%, 80.3%, and 80.3%, respectively, with the number of slots fixed at eight. Eight slots and 64 codebook entries are therefore used as the default configuration, providing a favorable accuracy–efficiency trade-off.

A4 Visualization of Recurrent Slot Refinement

To examine how the slot representation changes during recurrent refinement, we visualize the slot-grouping attention maps at each recurrent step under a 56×56 block occlusion. The first attention map is obtained when the stage features are initially grouped into slots. The slots then

Table A1: Ablation of the number of slots and codebook entries on ImageNet-1K. Only the examined hyperparameter is changed while all other configurations remain fixed.

Setting	Value	Top-1 (%)
Number of slots	4	79.8
	8	**80.3**
	16	80.2
Codebook size	32	80.0
	64	**80.3**
	128	80.3

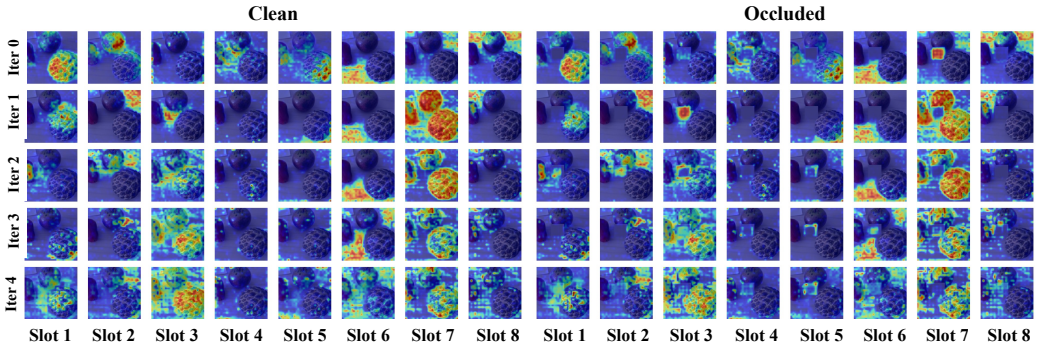


Figure A2: Visualization of slot-grouping attention across recurrent refinement steps. The clean-image attention is shown for reference. Under occlusion, the initial slot attention is fragmented and affected by the occluder. Repeated prototype retrieval and spatial redistribution progressively recover a spatial grouping closer to that obtained from the clean input.

pass through VQ retrieval and spatial redistribution, after which slot grouping is applied again to the updated spatial features to produce the attention map for the next recurrent step.

Figure A2 shows a representative example. At Stage 2, Slot 7 initially responds strongly to the visible lychee texture but also assigns substantial attention to the square occluder and nearby background. Consequently, its initial attention map is fragmented and differs clearly from the corresponding map obtained from the clean image. After repeated VQ retrieval and spatial redistribution, the attention gradually expands around the occluded region, becomes more concentrated on the lychee contour and texture, and suppresses the response to the occluder. By the later recurrent steps, its spatial pattern becomes substantially closer to the corresponding clean-image attention map. This behavior illustrates how recurrence progressively reorganizes incomplete spatial evidence into a more stable object-level grouping rather than repeatedly producing the same slot assignment.

A5 Role and Behavior of the VQ Prototype Memory

The codebook acts as a discrete memory of recurring visual patterns. For each continuous slot, the nearest codebook prototype is selected according to Euclidean distance. The retrieved prototype replaces the continuous slot and is subsequently redistributed to the spatial features through cross-attention, providing the representation used by the next recurrent iteration. Under incomplete observations, this mechanism allows an incomplete slot representation to retrieve related information from the learned prototype memory.

Codebook utilization. We measure prototype-utilization entropy over the ImageNet-1K validation set to examine whether the learned codebooks collapse to a small subset of entries. The utilization entropies for Stages 1–4 are 3.72, 3.58, 3.41, and 3.26, respectively. For a codebook containing 64 entries, the maximum entropy under uniform utilization is $\ln 64 = 4.16$. The relatively high entropy across all four stages indicates diverse prototype usage without severe codebook collapse.

Table A2: Prototype-utilization entropy at the four PRISM stages. The maximum entropy for 64 uniformly utilized prototypes is 4.16.

Stage	1	2	3	4
Utilization entropy	3.72	3.58	3.41	3.26

Table A3: Effect of VQ memory on ImageNet-1K classification under clean and occluded inputs.

Setting	Clean	$\text{PatchMask}_{0.8}$	BlockMask_{112}	OnlyOne_{112}
Full PRISM	**80.3**	**44.6**	**78.9**	**69.2**
w/o VQ at inference	80.2	37.8	75.6	62.7
Trained w/o VQ	80.0	37.2	75.2	63.1

Effect of removing VQ. We compare the full PRISM model with two settings that remove prototype memory: bypassing VQ only during inference and training PRISM from scratch without VQ. The results are summarized in **Table A3**. Removing VQ has little influence on clean-image accuracy, but produces considerably larger degradation when the visual input is incomplete. Moreover, training without VQ gives results similar to removing VQ only at inference. These observations indicate that the main contribution of the VQ memory is improved representation robustness under incomplete inputs rather than standard clean-image accuracy.

A potential concern of dataset-level prototype retrieval is that a mismatched prototype may introduce inaccurate information. The empirical robustness results suggest that this effect is limited in practice. If retrieval errors frequently accumulated across stages and recurrent iterations, performance would be expected to degrade more severely when the available visual evidence becomes increasingly incomplete. Instead, PRISM consistently improves over the feedforward PVTv2-B1 baseline across all evaluated occlusion conditions, as detailed in **Section A6**.

A6 Extended Robustness Evaluation

Performance across different masking ratios. We evaluate random patch masking at ratios of 0.2, 0.4, 0.6, and 0.8 to characterize performance degradation as the amount of available visual information decreases. The results are shown in **Table A4**. PRISM consistently outperforms PVTv2-B1 across all masking ratios. The difference becomes particularly pronounced under substantial information loss, demonstrating the benefit of recurrent prototype refinement for incomplete observations.

A7 Efficiency Analysis

The reported 2.2G FLOPs of PRISM account for the complete computation across all recurrent stages rather than a single recurrent pass. The average numbers of halting steps at Stages 1–4 on ImageNet-1K are 2.23, 2.60, 3.55, and 2.12, respectively, and these stage-wise averages are included in the FLOP calculation.

We further measure wall-clock efficiency on a single NVIDIA A6000 using 224×224 inputs, mixed-precision inference/training, and a batch size of 128. **Table A5** reports inference throughput, training throughput, and peak training memory under the same implementation and hardware environment.

Without VQ, PRISM remains slightly faster than PVTv2-B1 because the recurrent blocks share parameters and operate through lightweight slot representations. The complete model introduces additional prototype matching and redistribution, resulting in moderate inference overhead and a larger training overhead because activations from recurrent steps must be retained for back-propagation through time. Recurrent weight sharing nevertheless prevents the parameter count from increasing with the number of refinement steps.

Table A4: ImageNet-1K Top-1 accuracy (%) across different random patch masking ratios.

Model	0.2	0.4	0.6	0.8
PVTv2-B1	74.8	66.9	53.8	39.6
PRISM	**79.1**	**77.0**	**69.1**	**44.6**

Table A5: Practical efficiency measured on a single NVIDIA A6000 with 224×224 inputs, mixed precision, and batch size 128.

Model	Inference (images/s)	Training (images/s)	Peak Memory (GB)
PVTv2-B1	920	380	13.6
PRISM w/o VQ	**965**	**405**	14.1
PRISM	840	315	15.7

A8 Additional Comparisons and Positioning

Comparison with recurrent vision architectures. **Table A6** compares PRISM with representative recurrent or recursive vision architectures. SReT provides lightweight recursive computation but achieves substantially lower ImageNet-1K accuracy. CORnet-S also employs recurrent computation but is designed primarily as a computational model of the ventral visual pathway rather than as a high-performance general-purpose vision backbone. ClusterFormer-Tiny recurrently updates cluster centers through cross-attention and obtains slightly higher ImageNet-1K accuracy, but uses substantially more parameters than PRISM. PRISM provides a competitive accuracy–efficiency trade-off while additionally incorporating prototype memory and adaptive computation.

PRISM is only loosely inspired by the idea that human perception iteratively revisits and refines incomplete information and is not intended as a computational model of the human visual system. Unlike CORnet-S, PRISM does not explicitly map its stages to V1, V2, V4, and IT, model neural-response dynamics, or optimize for correspondence with brain recordings. Its recurrent design instead targets robust visual representation learning under incomplete observations.

Comparison with MAE. We also compare PRISM with the fine-tuned MAE ViT-B model, which is explicitly pretrained using masked-image reconstruction. As shown in **Table A7**, MAE achieves higher accuracy, particularly under random patch masking. This behavior is consistent with its masked reconstruction pretraining objective. However, MAE ViT-B contains approximately 86M parameters, compared with 11M for PRISM, and uses an 800-epoch self-supervised pretraining stage followed by supervised fine-tuning. PRISM is trained directly from scratch for 300 epochs using a standard classification objective, while maintaining competitive performance under both clean and occluded inputs.

Comparison with MetaSlot. PRISM and MetaSlot share the high-level idea of applying vector quantization to slot representations so that slots can be associated with learned prototypes, but the two methods differ in several important aspects.

First, MetaSlot applies slot quantization in a fixed feedforward procedure, whereas PRISM recurrently repeats slot grouping, prototype retrieval, and spatial redistribution within each stage. Adaptive Computation Time further determines the required number of iterations for each input, enabling more difficult or incomplete samples to receive additional refinement.

Second, MetaSlot introduces quantization between two Slot Attention aggregation stages, whereas PRISM integrates the slot–VQ–redistribution cycle into all four stages of the hierarchical backbone. Prototype memory can therefore refine both fine spatial representations and higher-level semantic representations.

Third, MetaSlot operates on pretrained visual representations such as DINOv2 features, whereas PRISM is trained end-to-end from raw images without requiring a pretrained encoder. PRISM

Table A6: Comparison with representative recurrent or recursive architectures on ImageNet-1K. "–" indicates that the corresponding value is not reported in the original work.

Model	Params (M)	FLOPs (G)	Top-1 (%)
SReT	4.0	0.7	74.0
CORnet-S	–	–	73.1
ClusterFormer-Tiny	27.85	–	**81.31**
PRISM	11.0	2.2	80.3

therefore functions as a standalone vision backbone rather than relying on semantic representations already encoded by a pretrained visual model.

Fourth, the two methods have different target objectives. MetaSlot primarily targets object-centric decomposition and adaptive slot allocation, whereas PRISM is designed to improve representation robustness when visual evidence is incomplete. Achieving this requires recurrence for progressive refinement and multi-level integration to address occlusion in both fine spatial details and high-level semantics.

Finally, the VQ modules are optimized differently. MetaSlot does not explicitly optimize the quantized representation to reconstruct the continuous input to the VQ module. PRISM uses reconstruction and commitment objectives to preserve the information contained in continuous slots during quantization. This reduces quantization distortion and makes the retrieved prototypes more suitable for repeated refinement across recurrent steps and hierarchical stages.

Table A7: Comparison with MAE ViT-B on clean and occluded ImageNet-1K inputs.

Model	Params (M)	Clean	PatchMask$_{0.6}$	PatchMask$_{0.8}$	BlockMask$_{112}$	OnlyOne$_{112}$	OnlyOne$_{56}$
MAE ViT-B	86	**83.6**	**71.4**	**48.7**	**80.2**	**71.1**	**35.6**
PRISM	11	80.3	69.1	44.6	78.9	69.2	33.8